APPLICATION OF ADAPTIVE NEURAL NETS (ANN) IN IRON AND STEELMAKING

A Thesis Submitted

in Partial Fulfillment of the Requirement

for the Degree of

MASTER OF TECHNOLOGY

by

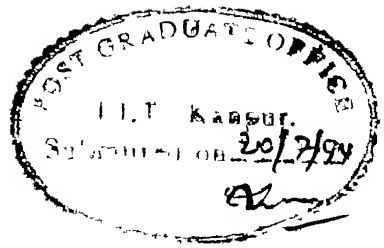
AMLAN DATTA

to the

DEPARTMENT OF MATERIALS AND METALLURGICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JULY 1994



CERTIFICATE

It is certified that the work contained in the thesis entitled "*Application of Adaptive Neural Nets (ANN) in Iron and Steelmaking*", by Amlan Datta, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Brahma Deo 20/7/94

Professor

Department of Materials and Metallurgical Engineering

Indian Institute of Technology Kanpur

July 1994

ABSTRACT

Neural nets can be adapted to complex patterns of interrelated input and output variables in a process even if the data sets contain a lot of noise. In this work two specific examples of the application of Adaptive Neural Net (ANN) in steel industry are described. First, the sulphur content of hot-metal, obtained at the end of calcium carbide powder injection in 400 ton torpedoes is predicted as a function of hot-metal weight, treatment time, initial sulphur content, gas flow rate and powder injection rate. The values predicted by the trained ANN model for a completely new set of input test data compare well with the actual values obtained on the shop floor. In the second example, the sulphur and phosphorus content of steel, obtained at the end of blow is predicted as function of liquid-metal weight, total amount of oxygen blown, amount of iron ore added, and the temperature, contents of carbon, manganese, phosphorus and sulphur determined by in-blow sampling in a 300 ton combined blown converter. The ANN predicted values of sulphur content of steel at tap (without reblow) also agree well with the values obtained on the shop floor. In these two cases net configuration and net parameters (learning rate, momentum factor and logistics) are determined by trial and error. Optimization of learning rate is achieved by Davies-Swann-Campey (DSC) search method and Golden-section search (GSS) method. The influences of preprocessing of data by dint of logarithmic transformation and Genetic Adaptive Search (GAS) are discussed. It is shown that if a mathematical relation between inputs and outputs is not available, then ANN model coupled with GSS and logarithmic transformation of data can be a good alternative for process modeling.

ACKNOWLEDGEMENT

I take the opportunity to express my heartfelt gratitude to my thesis supervisor Prof. Brahma Deo who had been inspiring me to work in the area which was alien to me. His able guidance and encouragement helped me to pursue the project.

I am thankful to Dr. Kalyanmoy Deb for sparing his precious time and giving me valuable suggestion, whenever I was lost in the labyrinth of Neural Networks.

I am grateful to Mr. Joydev Mukherjee and Mr. K.S. Rao for patiently letting me to get on their real nerves with my questions on Artificial Neural Network.

It is a nice opportunity to thank Mr. M. Hareesh who had originally started the work and made substantial progress.

I also thank Dr. Rob Boom of Corporate Research Laboratorium, Hoogovens IJmuiden of the Netherlands for supplying all the data used in this work.

I express my gratitude to my friends Amitava, Panu, Vidya, Dinesh, Sandip(da) and Indra(da).

A special thank is owed to my friend Sunita for typing a large portion of this thesis and helping in so many ways during the work.

A special mention is due to Mr. V.P. Gupta for making beautiful tracings of the figures with all his sincere effort and expertise.

Lastly I thank all those who has directly or indirectly helped me in I.I.T. Kanpur.

AMLAN DATTA

TABLE OF CONTENTS

Page

CHAPTER

List of Figures

List of Tables

List of Symbols

A	Constant which controls decay of output in Grossberg learning
A_f	Nominal area of slag-metal contact in torpedo vessel (m^2)
E	Error in prediction of ANN
f	Fraction of particles residing inside the bubble at the bubble-metal interface
I	Input to neuron 'j'
k_p	Mass transfer coefficient in metal around FRP (m/s)
k_g	Mass transfer coefficient of sulphur in metal for PIB (m/s)
k_f	Mass transfer coefficient of permanent reaction (m/s)
L	Partition coefficient
L_β	Width of search space in Golden-section method of optimization
m	Fraction of bubble base area contributing to desulphurization
n	Number of training patterns
Nu	Nusselt's No.
O_j	Output of neuron 'j'
O_B	Output of Bias neuron
Pr	Prandtl No.
Q_{stp}	Gas flow rate (m^3/s) at stp
Re	Rayleigh No.
S_i	Initial sulphur content of metal (wt%)
S_f	Final sulphur content of metal (wt%)
T	Logistics
t_k	Target output of neuron 'k' in the output layer
t_{rp}	Residence time of particles in plume(s)

t_{rb}	Residence time of bubble in the bath (s)
t_{in}	Injection time (s)
V	Volume of metal bath (m^3)
T	Temperature of bath (K)
W_{ij}	Weight of interconnects between neuron 'i' and 'j'
W_{Bj}	Weight of interconnects between Bias and neuron 'j'
α	Momentum factor
β	Learning rate
γ	Proportionality constant
δ	Error signal of neuron output
ϵ	Accuracy of search in Golden-section optimization
Ω	Surface of curvature
ρ_{sl}	Density of slag (kg/m^3)
ρ_s	Density of metal (kg/m^3)
θ_j	Threshold of neuron 'j'

Chapter 1
Introduction

1.1 Neural Network : Basic Elements and Terminology

The idea of neural network was originally conceived as an attempt to model biophysiology of the brain, i.e. to understand and explain how the brain operates and functions. The goal was to create a model capable of human thought processes. The brain is not constructed to think abstractly - it is constructed to ensure survival in the world and has many characteristics of a good engineering solution applied to mental operation : do as good a job as you can, cheaply and easily. The two mutually reinforcing objectives of neural modeling were defined and still remain today : first to understand the physiological and psychological functioning of human neural system; and second, to produce computational system.

The human nervous system is built of cells called *neurons*. It is known to be is of staggering complexity. An estimated 10^{11} neurons participate in perhaps 10^{15} interconnections over the transmission paths that could range for a meter or more. Each neuron is capable of receiving, processing and transmitting electrochemical signals over the neural paths that make up for the brain's communication [1].

Neural Networks are based on the biological nervous system. Each neuron in the brain is composed of a *body*, one *axon*, and a multitude of *dendrites* (Fig.1.1). The dendrites form a very fine 'filamentary brush' surrounding the body of the neuron. The axon is like a long thin tube which splits into branches terminating in little *endbulbs* almost touching the dendrites of other cells. The small gap between an endbulb and a dendrite is called a

synapse. The axon of a single neuron forms synaptic connections with many other neurons. Impulses propagate down the axon of a neuron and impinge upon the synapses, sending signals of various strengths down the dendrites of other neurons. Biological neurons are complex electrochemical devices that have a continuous internal potential, which is a function of inputs. If the internal potential exceeds a certain threshold then the neuron fires, that is, it effects other neurons by giving them its output, otherwise it is as good as a neuron with zero output. The strength of a given signal is determined by the efficiency of the synaptic transmission. A particular neuron will send an impulse down its axon if sufficient signals from other neurons impinge upon its dendrites in a short period of time, called the *period of latent summation.* The signal impinging upon a dendrite may be either *inhibitory* or *excitatory.* A neuron will send an impulse down its axon, if its excitation exceeds its inhibition by a critical amount called the *threshold* of the neuron.

1.1.1 General Model of Neuron

A general model of the basic processing element of Artificial Neural Network (ANN), or connectionist architecture, can be considered to have following three elements [2].

a) Weighted Summing Unit

This accepts external/internal inputs $I_i = [I_1,\ldots,I_n]$ multiplied by a corresponding weights $W_i = [W_{i1},\ldots,W_{in}]$ and fixed weighted input θ_i (bias or threshold) and provides a summed output p_i. The external inputs may be either from environment or system to which ANN is being applied, whereas the internal inputs may be either from previous layer

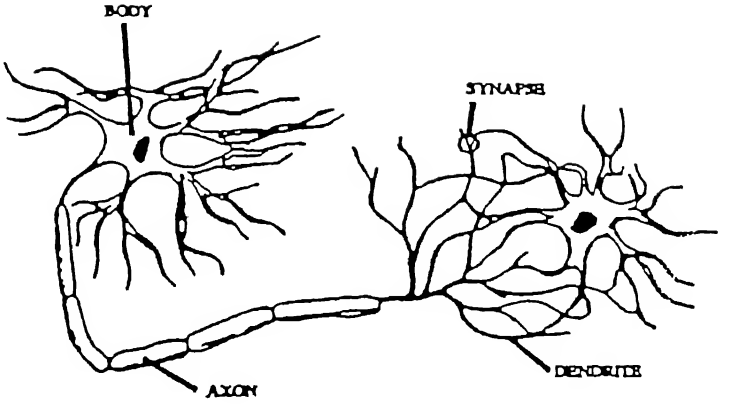


Figure 1 1 A Biological Neuron [1]

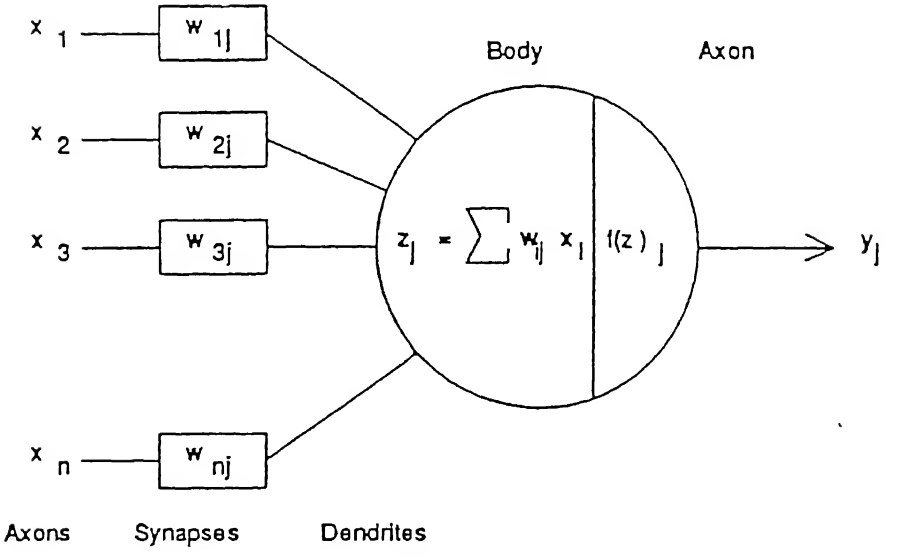


Figure 1.2 The McCulloch-Pitts Neuron [1]

of ANN or the outputs of neurons. In case these inputs are derived from neuron outputs, it forms a feedback architecture. Else, it has a feed-forward architecture.

b) Linear dynamical function

It is essentially a single input single output function block. This block may exist for time varying signals and introduces a function which is either a integral, a proportional, a time delay or a combination of them. For example, the following two general functions can be used to relate input I_i with output O_i as

$$a_1 O_i(t) + a_2 O_i(t) = I_i(t) \tag{1.1}$$

$$O_i(t) = I_i(t - T) \tag{1.2}$$

where a_1, a_2 are constants and T is the time delay.

c) Nonlinear function

This function decides the firing of neuron for given input values. It is a nondynamic (static) nonlinear function which may be pulse type or step type, differentiable (smooth) or nondifferentiable (sharp) and having either positive mean or zero mean. Some of the examples of such functions are threshold, sigmoid , tanhyperbolic or Gaussian functions.

One can obtain different characteristics of neurons with different types and combination of the above mentioned basic components. For example a perceptron model is made up of weighted summing unit having no feedback input, no dynamic function and sigmoid as non-linear function, while feedback or dynamic networks utilize the dynamic function block.

Fig.(1.2) shows an 'artificial' neuron, which models the behaviour of the biological neu-

ron. The body of the jth neuron will be represented by a weighted linear sum z_j, of the input signals followed by a linear or nonlinear function,

$$y_j = f(z_j) \tag{1.3}$$

The function $f(z_j)$ is called an activation function, a function which uses the input value to determine the output activity of the neuron. This neuron description is termed as McCulloch-Pitts neuron. A McCulloch-Pitts type neuron using binary threshold is referred to as a perceptron [1]. Different neural networks use different functions $f(z_j)$, but the internal structure of the neuron i.e. linear sum followed by a function $f(z_j)$, is common to most networks. The synaptic efficiencies will be represented by the *interconnection weights*, w_{ij}, from the ith neuron to the jth neuron. The weights can be either positive(excitatory) or negative(inhibitory).

1.1.2 Learning, Recall and Memory in ANN [2]

The development of any Artificial Neural Network(ANN) involves two phases. First the *learning* or the *training* phase and the second *recall* or *testing phase*. Memory, in ANN, is in the forms of values of weights of the interconnecting links. The memory in ANN can be a Content Addressable Memory(CAM), where it stores data at stable states in memory(or weight) matrix W or an Associative Memory which provides output response from input stimuli.

The mechanism for learning alters the weights associated with the various interconnections and thus leads to a modification in the strength of interconnection. Training is

carried out by presenting the network examples called training patterns. Once the network has learnt the problem it may be presented with new unknown patterns and its efficiency can be checked. This is called testing phase.

A network is trained so that an application of a set of inputs produces the desired (or at least consistent) set of outputs. Each such input (or output) set is referred to as a vector. Training is accomplished by sequentially applying input vectors, while adjusting network weights according to a predetermined procedure. During training, the network weights gradually converge to values such that each input vector produces the desired output vector.

Neural networks will either have fixed weights or adaptable weights. Networks with adaptable weights use learning laws to adjust the value of the interconnection strengths. If the neural network is to use fixed weights, then the task to be accomplished must be well defined a *priori.*

All learning methods can be classified into two categories : supervised and unsupervised.

a) Supervised learning

Supervised learning requires the pairing of each input vector with a target vector representing the desired output; together these are called a *training pair.* Usually a network is trained over a number of such training pairs. An input is applied, the output of the network is calculated and compared to the corresponding target vector, and the difference(error) is fed back through the network and weights are changed according to an algorithm that tends to minimize the error. The vectors of the training set are applied sequentially, and errors are calculated and weights adjusted for each vector, until the error for the entire training set reduces to an acceptably low level.

b) Unsupervised learning

Supervised learning has been criticized as being biologically implausible because it is difficult to conceive of a training mechanism in the brain that compares desired and actual outputs, feeding processed corrections back through the network.

Unsupervised training is a far more plausible model of learning in the biological system. Developed by Kohonen(1984) and many others, it requires no target vector for the outputs, and hence, no comparisons to predetermined ideal responses. The training set consists solely of input vectors. The training algorithm modifies network weights to produce output vectors that are consistent; that is, both application of a vector that is sufficiently similar to it will produce the same pattern of outputs. The training process, therefore , extracts the statistical properties of the training set and group similar vectors into classes. Applying a vector from a given class to the input will produce a specific output vector.

In all the cases *Learning laws* determine how the network will adjust its weights while using an error as function of weights or some other criteria.

1.1.3 Classification of ANN

Depending on the neuron interaction, neural nets can be classified as-

a) Feed-back Neural Network and b) Feed-forward Neural Network

a) Feed-back Neural network

Here, neurons are arranged in the form of undirected graphs. Also called Recurrent

networks, these recirculate previous outputs back to inputs; hence their outputs are determined both by their current inputs and their previous outputs. For this reason recurrent networks can exhibit properties very similar to short-term memory in humans in that the state of the network outputs depends in part upon their previous inputs. The connections in this case are symmetrical and bidirectional. Feedback ANN models use discrete computation. Here the system is initialized and then it evolves to a final state in the course of time. The stability of this neural net is analyzed with the help of energy functions that can be defined in the term of states or neurons, weights and thresholds. Under certain conditions, the energy function decreases monotonically as the network moves from one state to another. The stability and convergence can be analyzed by studying the descent of scalar energy functions instead of the transition of states.

1) Learning in unsupervised nets by Grossberg learning [1]

D.O. Hebb(1961) proposed a model for unsupervised learning. Hebb's law states "If a neuron, X is repeatedly stimulated by another neuron Y, at times where neuron X is active for whatever reason, then neuron X will become more sensitive to stimuli from neuron Y; the synaptic connection from Y to X will be more efficient. Thus, Y will find easier to stimulate X in future.". According to this law synaptic strength(weight) was increased if both source(input cell) and destination neuron(output) cell were activated. In this way, the often used paths in the network are strengthened and the phenomena of habit and learning through repetition are explained. If input cell i and the output cell j are activated, then the change in weights for the interconnection between i and j is given by

$$\Delta W_{ij} = \beta O_i O_j \qquad 1.4$$

where

β = learning rate parameter

O_i activates input i

O_j activates output j

If we start from zero initial weights, then after training

$$W_{ij} = \sum_{n=1}^{N} O_{in} O_{jn} \qquad 1.5$$

The W_{ij} are proportional to the correlation between units i and j. These weights are usually normalized. Once the weights are learnt the system is used to predict the output for a new input.

Hebb's law is not a mathematical statement that directly applies to work with neural networks. Grossberg learning is an attempt to mathematically formulate Hebb's law and to explain the classical conditioning behaviour of neuron. The neuron proposed by Grossberg uses inputs $O_j(t)$ from other neurons of ANN as well as external inputs $I_j(t)$. If W_{ij} is the weight connecting the output of the ith neuron to the input of the jth neuron and A is a positive constant controlling the decay of the output in absence of any other inputs, then dynamics of the jth neuron having output $O_j(t)$ can be written as (n is the total number of neurons)

$$\frac{\partial O_j(t)}{\partial t} = -AO_j(t) + I_j(t) + \sum_{i=1}^{n} W_{ij} O_j(t) \qquad 1.6$$

Above equation contains the capability for the self feedback if $W_{ii} \neq 0$. Consider the equation which controls the learning of the network. The weight from the ith neuron to the jth neuron is adapted according to

$$\frac{\partial W_{ij}(t)}{\partial t} = -FW_{ij}(t) + GO_j(t)O_i(t) \qquad 1.7$$

where F and G are positive constants, and $O_j(t)$ is the output of the jth neuron. The first term on the right hand side of the Eqn.(1.7) allows the neuron to forget a relationship over time, with F controlling the forgetting rate. This constant has a much smaller value than A. The second term is to implement Hebb's law, with G controlling the learning rate. If both the input signal and the output signal are large then the weight change will be large. If either signal is small then the weight change will be small.

2) Learning in unsupervised nets by Kohonen Model [1]

T. Kohonen in early 1981 introduced self-organizing network (Kohonen maps) which uses unsupervised learning and organizes itself to topologically represent characteristics of the input patterns. It is a fully interconnected array of neurons i.e. the output of each neuron is an input to all of the other inputs in the network, including itself. Any of the interconnection strengths could be zero. Each neuron has two set of weights : one set is used to compute the sum of weighted external inputs and another to control the interaction between neurons in the network. The weights on the input pattern are adjustable, while the weights between neurons are fixed. Each neuron receives an identical input pattern, A, consisting of n elements. A diagram of a simple Kohonen network with N neurons is shown in Fig.(1.3).

During each presentation, the complete input pattern is presented to each neuron and each neuron computes its output as a sigmoidal function on the sum of its weighted inputs. The input pattern is then removed and the neurons interact with each other to locate the neuron with the largest output and to allow only that neuron to produce output. The interaction also isolates a neighbourhood of that neuron, and allows only that neuron and its neighbours to adjust their input weights. The adjustment is done such that the neuron

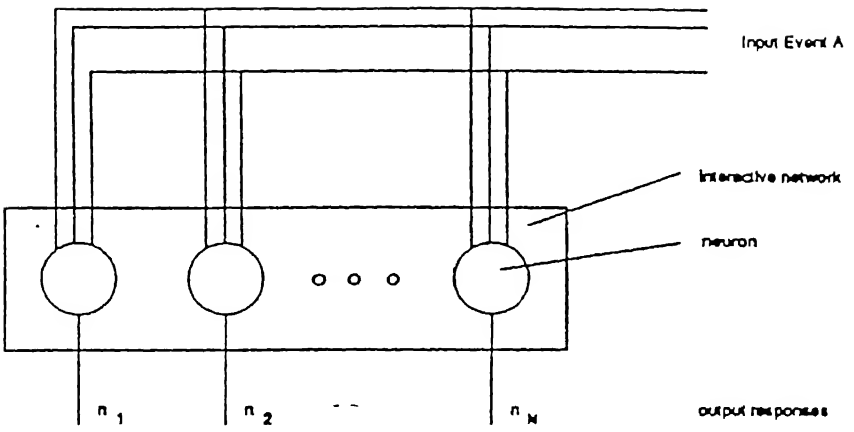


Figure 1 3 Kohonen Network [1]

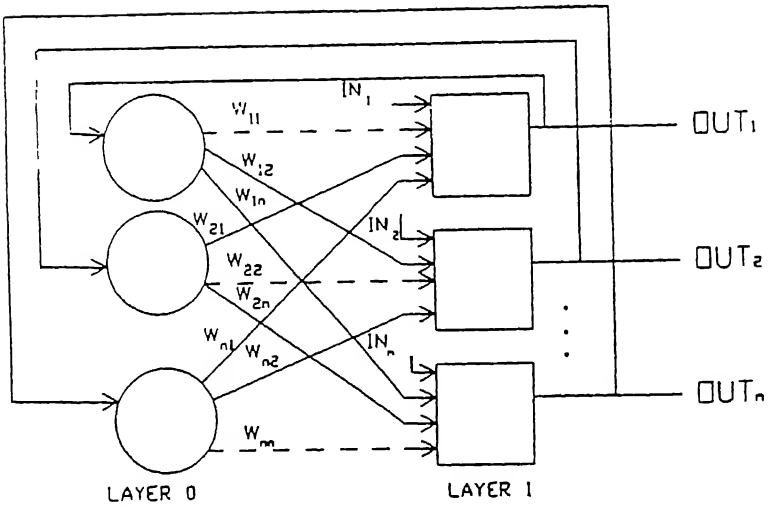


Figure 1 4 Recurrent Network [3]

adjusting their input weights would have a larger output if the exact same input pattern was presented again. This network may consist of several modules and can be used for pattern recognition, robotics, process control and even for processing of semantic informations.

3) Learning in unsupervised nets by Hopfield Model

Hopfield nets, named after its inventor John Hopfield [1982] [3], have lateral and recurrent connections, that is, the output of a neuron is fed back to itself and the intra-layer connections are present.

Fig.(1.4) shows a recurrent network consisting of two layers. The format is functionally equivalent to the Hopfield model.

Layer 0 serves no computational function; it simply distribute the network outputs back to its inputs. Each layer 1 neuron computes the weighted sum of its inputs, producing a net input (I) signal that is operated on by the nonlinear function to yield the output (O) signal.

In Hopfield's early work [1982] [3], the function was a simple threshold. The output of such a neuron is 1.0 if the weighted sum of the outputs of other neurons is greater than a threshold θ_j; otherwise it is 0. The connection weights are set according to the Hebbian principle [4]. The output is calculated in the following manner [3].

$$I_j = \sum_{i \neq j} W_{ij} O_i + IN_j \qquad 1.8$$

O_j = 1 if $I_j > \theta_j$

O_j = 0 if $I_j < \theta_j$

O_j unchanged if $I_j = \theta_j$

The *state of a network* is simply the set of the current values of the output signals from all neurons. In the original Hopfield network, the state of each neuron changed at discrete random times; in later work, the neuron states could change simultaneously. Because the output of a binary neuron can be only 1.0 or 0.0, the current state of the network forms a binary number, each bit of which represents the output signal from a neuron. The state of the network is said to be unstable if it keeps on oscillating from one state to another.

Any change in the state of a neuron will either reduce the energy or maintain its current value. Because the energy shows this continuous downward trend, eventually it must find a minimum and stop. Thus stable configurations achieve permanent state after a finite number of changes. Thus the network is stable.

The learning is unsupervised and takes place off-line. Hopfield nets can be used as associative memories. They can also be used to solve optimization problems. They give better results when the input is perfectly represented as a string of binary bits(e.g. ASCII characters shown as a string of 8 binary bits). Hopfield nets suffer from two major limitations [4]. Firstly, not more than 0.15N number of patterns can be stored on a net, N being the number of nodes in it. Secondly, an exemplar pattern will be unstable if it shares many bits in common with another exemplar pattern. Here an exemplar is said to be unstable if it is applied at time zero, and the net converges to some other exemplar.

b) Feed-forward Neural Networks

In feed-forward neural networks the neurons are arranged in layers like directed graphs. Inputs are applied to the layers and outputs are collected. Stability is not a problem here because these networks are loop free. The computation time in such neural nets is the time required for the signals to propagate and the output to settle down. Also called as

nonrecurrent network, in these networks the output of any given neuron can not be fed back to itself directly or indirectly or through other neurons, and so its present output(s) does not the influence future outputs. In these networks computations are completed in a single pass. Each layer receives input values from the previous layer, computes output values and passes these values onto the next layer. When the output values of the final layer are determined the computation ends. This rule can be broken only during the training or learning phase, when the output of a neuron can be used to adjust its weights, thus influencing future outputs of the neuron. This can be further classified in following way.

1) Learning in supervised nets by Grandmother Cells [1]

Grandmother cells (Fig.1.5) are comprised of neurons with no interconnections. The input weights are fixed, and each neuron simply computes a linear sum of the weighted external inputs. Each neuron receives the same input but has different weights. Input pattern and weights are treated as vectors, then the grandmother cell computes the dot product of the input vector with the weight vector. The output of the cell indicates how 'closely' the input vector matches the weight vector. If the input vectors and weight vectors are normalized to a length of one, then the output lies between +1 and -1, +1 corresponding to an exact match and 0 indicating that the vectors are orthogonal. The weights for the grandmother cell are based upon the pattern that the cell should respond to. An input pattern is presented to the network, each cell computes its output, and the cell with the largest output indicates which stored pattern the input pattern most closely resembles. Each input pattern is simultaneously compared with numerous test patterns. This concept is crucial to the use of neural networks for problems such as pattern recognition.

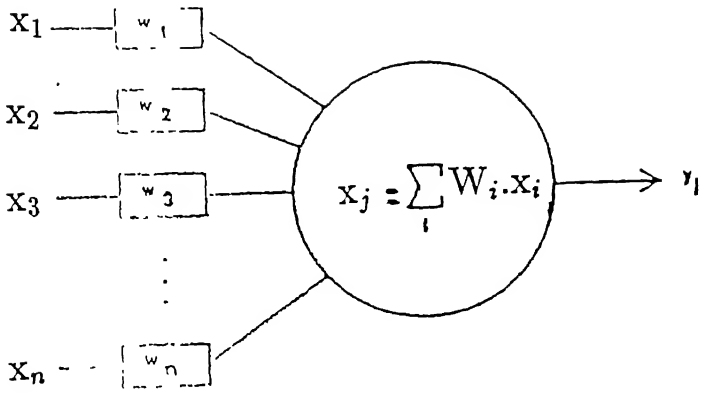


Figure 1 5 Grandmother Cell [1]

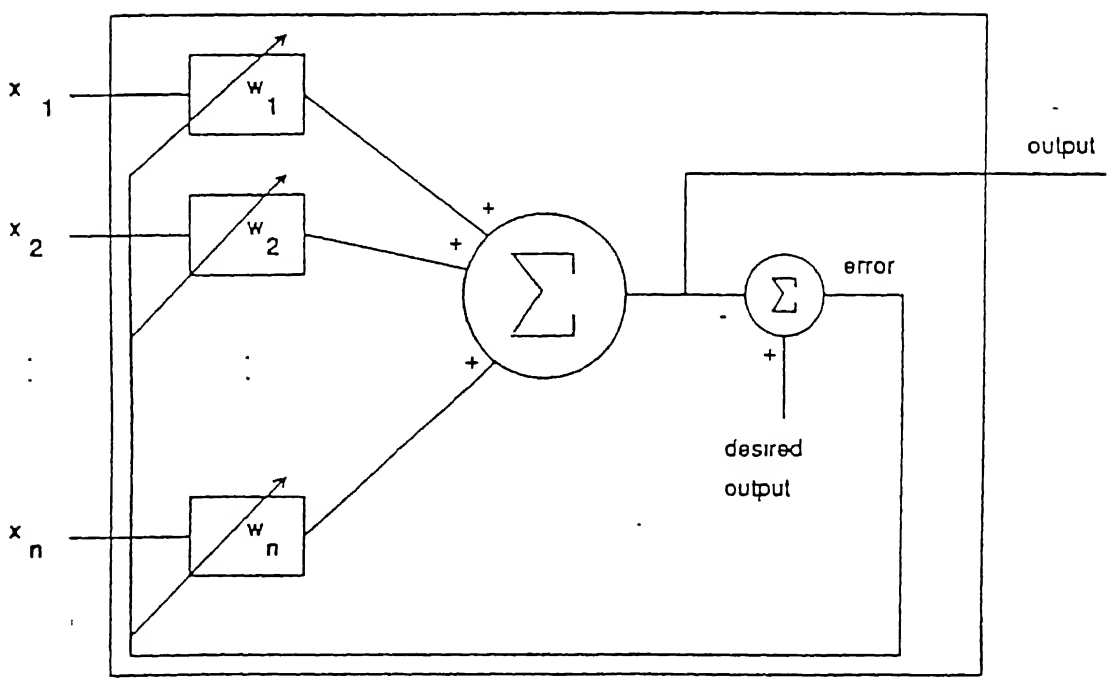


Figure 1.6 ADALINE [1]

2) Learning in supervised nets by ADALINE [1]

Widrow and Hoff used an ADALINE (ADAptive LINear Element) to associate an input pattern with an output value. An ADALINE, consists of a single neuron of the McCulloch-Pitts type with $f(z_j) = z_j$ and with weights determined by the normalized LMS (Least Mean Square) learning law. During training with supervised learning, the neuron computes an error signal, the difference between the desired output and the computed output, and then adjusts the weights so that the computed output is closer to the desired output. During the recall phase an ADALINE functions in the same fashion as a grandmother cell. Since ADALINE is a linear device, hierarchical layers of ADALINEs will not increase computational capability. Any linear combination of linear units can be accomplished with the use of a single linear unit.

The normalized Least Mean Square (LMS) algorithm attempts to minimize the mean squared error. There is a mean squared error value associated with every weight vector, given some statistics for the input signal. The set of these values form a performance surface in *weight space*. Gradient descent method is used to search for a minimum in which updates are made in the direction of the locally steepest descent. Usually, this means computing the derivative of the error surface at the current location. The ADALINE will always adapt to produce the minimum mean squared error, although when operating in the presence of noise this minimum error value is greater than zero.

Fig.(1.6) shows the structure of a single ADALINE. An input pattern is presented to the neuron, the neuron computes a weighted sum of the inputs and then computes the error signal. The error signal is used to adjust the weights using the Delta rule.

An ADALINE is usually used to make binary decisions, so the output is sent through a

binary threshold function (Fig. 1.7). This new device is an example of a perceptron and is only capable of performing binary transformation that are linearly separable. A problem is linearly separable if the input patterns plotted in the input space can be grouped as desired by bisecting the input space with a single hyperplane, e.g. a single straight line in two dimensional space. The classic example of a mapping that is not linearly separable is the XOR function. Assuming that each input is either +1 or -1 Fig.(1.8) shows the location of the four input patterns in input space and a symbol corresponding to the output of the XOR function for each. There is no way to draw a single straight line so that the circles are on one side of the line and the squares are on the other, i.e. the groups can not be linearly separated.

The next logical extension is to place perceptrons in a hierarchical structure. A layer of perceptrons may be connected to fixed logic gates to form solutions to nonlinearly separable problems. This type of network is called **MADALINE** (Many ADALINEs). In a MADALINE only the weights of the ADALINEs are adjusted. There are methods available to adjust the weights of the ADALINEs to solve specific problems using a MADALINE network. Fig.(1.9) shows a network with two perceptrons connected to a logical AND gate. When suitable weights are chosen this network is capable of implementing the XOR function, Fig.(1.10).

3) Learning in supervised nets by Backpropagation

The backpropagation network is an extension of the MADALINE structure involving the addition of a nonlinear *function* on the output of each neuron, and use of multiple adaptable layers. The algorithm originally developed by Rumelhart [5]. The weights are adaptable in all layers. The backpropagation network is capable of approximating as closely

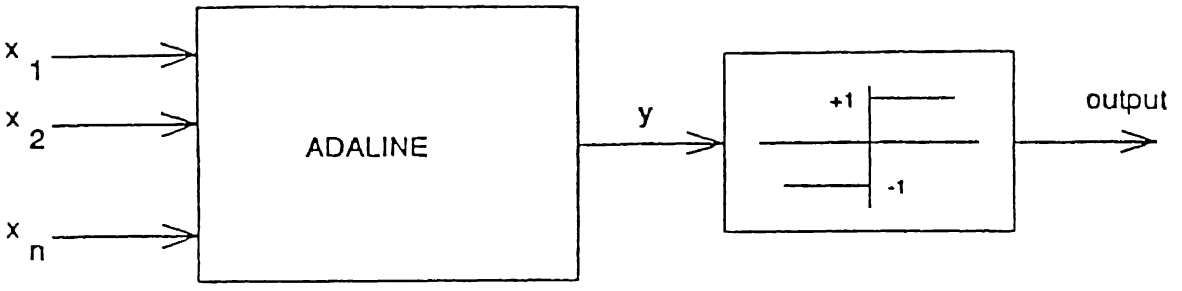


Figure 1 7 Perceptron [1]

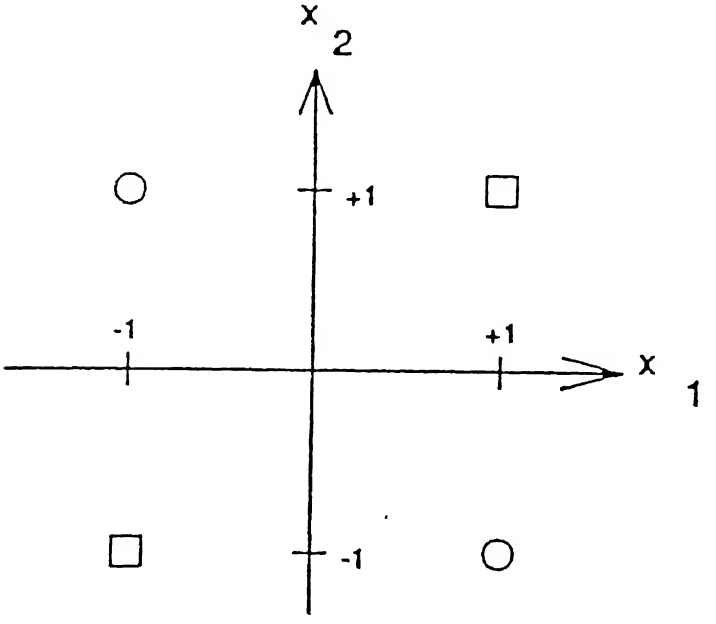


Figure 1.8 XOR Function [1]

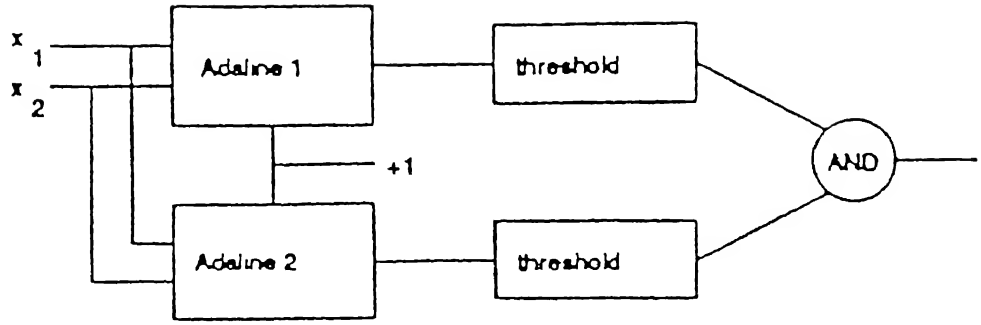


Figure 1 9 MADALINE [1]

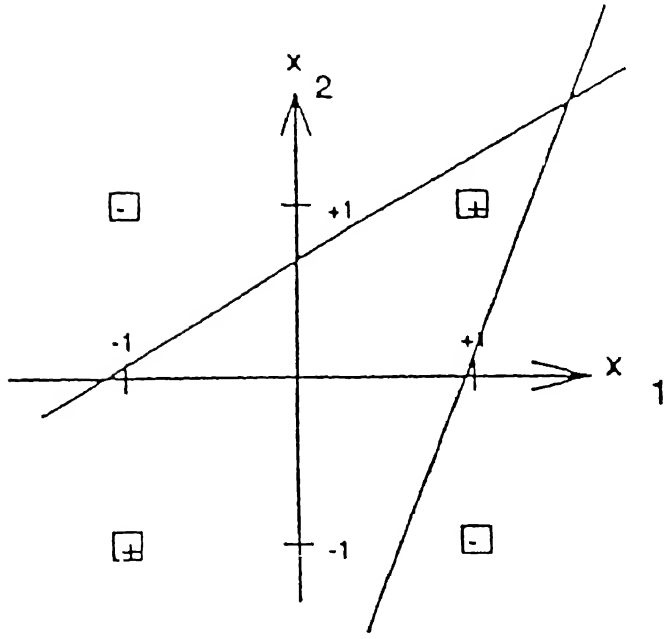


Figure 1.10 MADALINE applied on XOR [1]

as desired any mapping between any vector of dimension m and any other vector of dimension n. The algorithm for training the ANN can be understood with the help of a simple three layer perceptron model (Fig. 1.11). These layers are namely input layer (I), hidden layer (II) and the output layer (III). All the input signals to any neuron are multiplied by the weights of the connection with the neuron from which the input is received. The summation of all these weighted inputs provide the net input to the node. From hidden layer onwards a bias is added to all the neurons and this bias has a constant output of unity and it does not have any connection with neurons of the previous layer. The weights of bias neuron are also trained in the same manner like the other neurons. The net input to any 'j th' neuron of any layer can be written as

$$I_j = \Sigma O_i W_{ij} + W_{Bj} O_B \qquad 1.9$$

where

i = runs over all the neurons in the previous layer.

j = runs over any of the neuron of any layer

W_{ij} = Weight of the interconnections between i & j

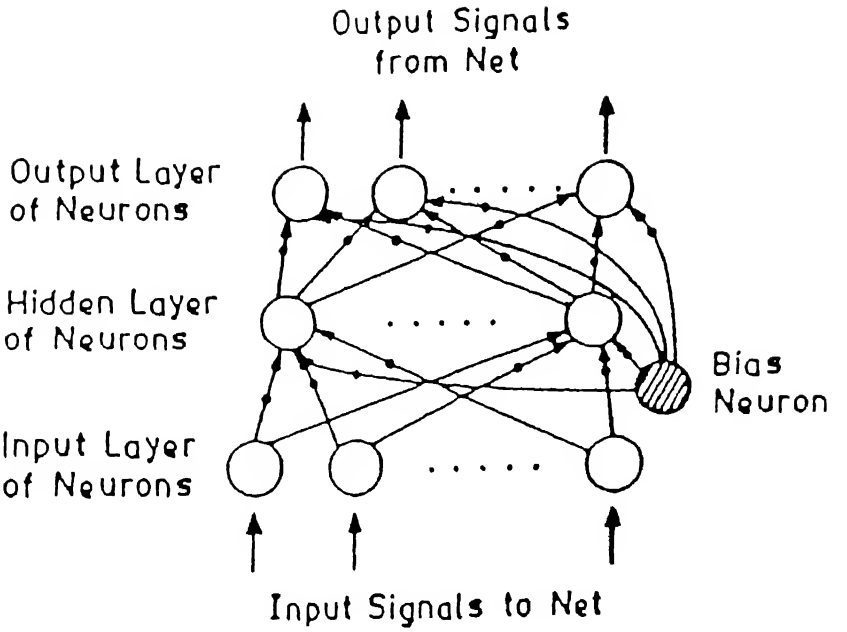
O_i = Output of i th neuron

O_B = is the output of bias, here it is 1.0

W_{Bj} = the weight of connection between bias & j

One can assume bias neuron as a member of the set of the neurons contributing to the input of a particular node of a layer. Thus Eqn.(1.9) reduces to

$$I_j = \Sigma O_i W_{ij} \qquad 1.10$$



- Connection Weights

Figure 1.11 Three layer ANN model.

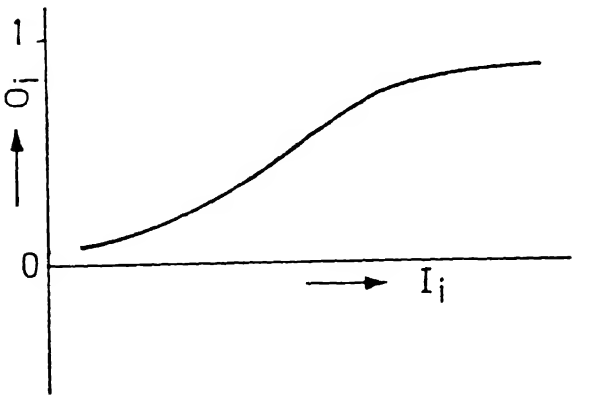


Figure 1.12 Relation between input and output of a neuron.

Depending upon the input, each neuron fires an output signal which follows the sigmoidal relationship. In other words the output of 'j' is given by

$$O_j = \frac{1}{1 + exp(-I_j/T)} \qquad 1.11$$

where T is logistics and decides the slope of the linear portion of the sigmoidal curve (Fig. 1.12). The error of learning is given by

$$E = \sum_k (t_k - O_k)^2 \qquad 1.12$$

where

k = all the neurons in output layer

O_k = is the predicted output by net.

t_k = is the scaled target output.

The error is considered to be a function of the weights. The weights are modified to obtain the lowest value of the error. The purpose of the training is to find out those interconnect weights for which the gradient of the error surface in the multidimensional weight space is almost zero. The modification of the weights occur in the negative proportion of the derivative of the error with respect to the weights. This method is known as **'Steepest Descent method'**. Thus

$$\Delta W_{ij} \alpha - \frac{\partial E}{\partial W_{ji}} \qquad 1.13$$

From Eqn.(1.12)

$$\frac{\partial E}{\partial W_{ij}} = -2[\sum_{k=1}^{n} (O_k - t_k)] \frac{\partial O_i}{\partial W_{ji}} \qquad 1.14$$

On differentiating Eqn.(1.11) with respect to weights

$$\frac{\partial O_j}{\partial W_{ij}} = O_j(1 - O_j) \frac{dI_j}{dW_{ij}} \qquad 1.15$$

From Eqn.(1.10) we obtain

$$\frac{\partial I_j}{\partial W_{ij}} = O_i \qquad 1.16$$

From Eqns (1.14),(1.15) and (1.16) we get

$$\frac{\partial E}{\partial W_{ij}} = -2(O_j - t_j)O_j(1 - O_j)O_i \qquad 1.17$$

From Eqn.(1.12) the change in weights are given by

$$\Delta W_{ij} = -\gamma \frac{dE}{dW_{ij}} \qquad 1.18$$

where γ is proportionality constant. Here the negative sign appears in the equation because the direction of steepest descent along the gradient has been approached. Now combining Eqns (1.17) and (1.18)

$$\Delta W_{ij} = 2\gamma(O_j - t_j)O_j(1 - O_j)O_i = \beta\delta_j O_i \qquad 1.19$$

where

β = learning rate

δ_j = error signal = $(O_j\text{-}t_j)(1\text{-}O_j)O_i$

To make a tentatively correct search in the weight space and thus make a better learning, a momentum factor α is incorporated in Eqn. (1.19) The momentum factor when multiplied by Δ W_{ij} adds a fraction of weight change of the previous iteration to the learning algorithm. Thus Eqn.(1.19) modifies to

$$\Delta_{n+1}W_{ij} = \beta\delta_j O_i + \alpha\Delta_n W_{ij} \qquad 1.20$$

where

$\Delta_{n+1}W_{ji}$ = change in the weights in $(n+1)^{th}$ iteration

α = momentum factor

Since we know the target output we can easily calculate the error signal of the nodes of output layer. But for hidden neuron target for each node is not available specifically. So the error signal of all the neuron is determined by the following expression

$$\delta_j = O_j(1 - O_j)\sum \delta_k W_{kj} \qquad 1.21$$

where k runs over all the neurons of the immediately succeeding layer. The next step after training is testing. A standard set of the input is passed through the trained net whose weights have been adapted according to the training patterns. In the testing phase, only forward pass is taken. The result of net predictions are compared with the actual outputs and when the desirable tolerance is achieved, the training stops.

1.2 The Application of ANN in Metallurgical Processes

In the last few years ANN has been applied to various metallurgical processes including welding, heat-transfer modeling, simulation of metal-slag equilibrium, control of electric arc furnace operation etc. These applications are briefly reviewed below.

1.2.1 Application of ANN in Welding

The arc welding processes are substantially nonlinear, in addition to their highly coupled multivariable nature. A finished weld can be characterized by a set of direct welding parameters (DWP) and indirect welding parameters (IWP). The former includes the pen-

etration of the weld pool, the bead width, the reinforcement height and the transverse cross-sectional area and the later includes welding voltage, current, torch travel speed, wire feed rate, electrode tip angle and shielding gas type and flow rate. Making a simultaneous correlation of DWP and IWP is not always possible. For example, while bead width can be monitored when welding is in progress, penetration is virtually unknown on-line.

Anderson *et.al.* [6] have developed an ANN model to simulate Gas Tungsten Arc Welding (GTAW) in which the DWPs are derived from the IWP's and vice-versa. The first application was on weld pool modeling where the neural network derives geometrical attributes of the weld pool from the parameters of the welding equipment. The second application was essentially a mean to determine the equipment properties to obtain a specific pool geometry. Training parameters can be selected randomly from a set of practical data.

In usual practice IWPs are decided by welder's previous experience and handbook recommendation. Some fine tuning is done by the operator. Obviously a lot of human error is involved in the whole procedure. The same approach can be accomplished by ANN in a more accurate way. A pictorial explanation of the scheme is given in Fig.(1.13). For a predecided set of DWPs a properly trained network produces required set of IWPs. This route of selection of IWPs requires minimal experimentation and experience to achieve a desired set of DWPs. In order to update the ANN model for a new condition the IWPs may be recorded, and similarly the resulting DWPs are also stored in a recorder after welding. Once these new data are available the neural network can be refined by additional off-line training with the new data, and thus its characteristics can be continuously updated for future application.

Closed loop control can also be achieved by ANN. An approach to close-loop control using neural network is illustrated in Fig.(1.14). As described before the welding equipment

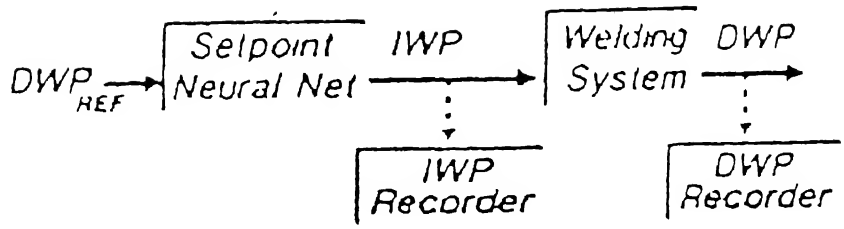


Figure 1.13 Indirect weld parameter selector using ANN [6]

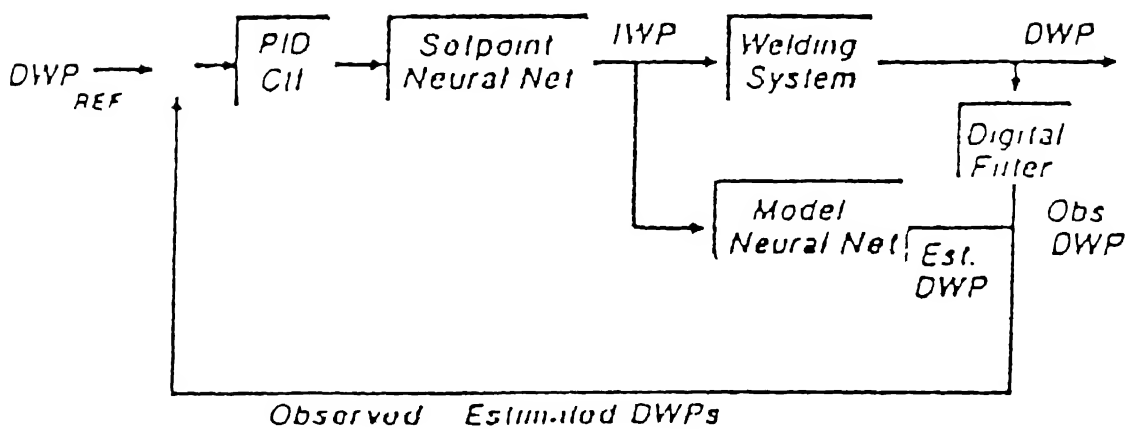


Figure 1.14 Closed-loop control in welding using ANN [6]

is tuned with respect to a set of IWPs determined by a off-line trained network. Parallel to the actual welding hardware a second ANN is fed with the same IWPs. This network is trained off-line to model the welding process. As a result it creates its own set of DWPs, which are estimates of actual DWPs produced by the welding process. From both these values of DWPs an on-line estimation of the entire DWP vector is obtained. This vector is then compared with the set point reference and an error vector is produced. This vector is fed to a PID controller, which in turn provides the control signal to the neural network that decides the set-point.

In a weld simulation study [6], actual GTAW data were used by the Anderson *et. al.*. The data conisted of four IWPs, namely welding current and voltage, welding speed and wire feeding speed as inputs and the four DWPs like weld reinforcement height, penetration of weld pool, bead width and transverse cross-sectional area of the weld were the outputs. The net was trained by 31 sets of data while 11 different sets were used for testing the net. From the description of the problem it is clear that the each of the input and the output layer comprises of 4 nodes. The configuration of the hidden layer was found out by trial and error method. Training was terminated when a predefined value of sum of the squared error was reached (0.09 as the limit of convergence). During training the correction gain or learning rate was kept constant at 0.1. The net performance was quite satisfactory during testing.

1.2.2 Application of ANN in Heat-transfer

The solution of heat-transfer problems in practical/industrial situation often involves the solution of complicated partial differential equation. Thibault *et. al.*[7] simulated by

artificial neural network (ANN). They chose three examples to demonstrate the methodology of neural networks for correlating the heat-transfer data viz thermocouple look-up table, correlation between Nusselt and Rayleigh numbers for the free convection around a horizontal smooth cylinders and heat-transfer by natural convection along a slender vertical cylinder with variable surface heat flux. Two types of learning algorithms incorporating gradient descent method and Quasi-Newton learning algorithm were tested.

a) Thermocouple look-up table

An ANN model was developed to give temperature corresponding to the electromotive force (EMF) generated by a chromel-alumel (type K) thermocouple. This simple problem was purposely chosen by the authors to show some important character istics of artificial neural network. Since bias neuron was also used in this particular case, the number of neurons or nodes in input and output layers were 2 and 1 respectively. According to the author there is no method available, which can accurately determine the number of hidden layers and corresponding number of nodes. The net was trained with 201 values of EMF within the temperature range of 0-200°C. It was observed that Quasi-Newton learning performed better than the gradient descent method and also gave a faster convergence. The tuning parameters , namely learning rate and momentum factor may not be necessarily kept constant but may be varied throughout the learning process.

b) Free convection around a horizontal smooth cylinder

The correlation between Nusselt number and Rayleigh numbers was established for horizontal smooth cylinders. Theoretically, these two quantities can be related by five empirical equations over a range of Rayleigh numbers. They are as follows

$$Nu = 0.675Ra^{0.058} \qquad 10^{-10} \leq Ra \leq 10^{-2} \qquad 1.22$$

$$Nu = 1.020Ra^{0.148} \qquad 10^{-2} \leq Ra \leq 10^{2} \qquad 1.23$$

$$Nu = 0.850Ra^{0.188} \qquad 10^{2} \leq Ra \leq 10^{4} \qquad 1.24$$

$$Nu = 0.480Ra^{0.250} \qquad 10^{4} \leq Ra \leq 10^{7} \qquad 1.25$$

$$Nu = 0.125Ra^{0.333} \qquad 10^{7} \leq Ra \leq 10^{12} \qquad 1.26$$

Here again the number of input nodes were 2(including bias node) and output node was 1. The neural net consisted of five hidden neuron in a single layer. Fig.(1.15) shows the plot of Nu as a function of Ra for the prediction of data that were generated with the set of five equations (Eqns 1.22-1.26) using a neural network. The ANN model gave an accurate representation of the generated data over a complete range of Rayleigh numbers. It is clear that a unique neural model can adequately replace the set of five equations. One specific aspect of their study was that both the dimensionless numbers were introduced to the ANN after a logarithmic transformation (Fig.1.15). However the logarithmic transformation preprocessing of data may not be always useful, because the behaviour of the net depends upon the nature of the training and testing data as well.

c) Natural convection along a slender vertical cylinder

Attempt was made to correlate the local and the average Nusselt number (Nu) as a function of the Prandtl number (Pr), the exponent of the power law variation of the surface heat flux (n) and the surface curvature (Ω) to describe the natural convection along slender vertical cylinder with variable heat fluxes. In literature the same correlation is

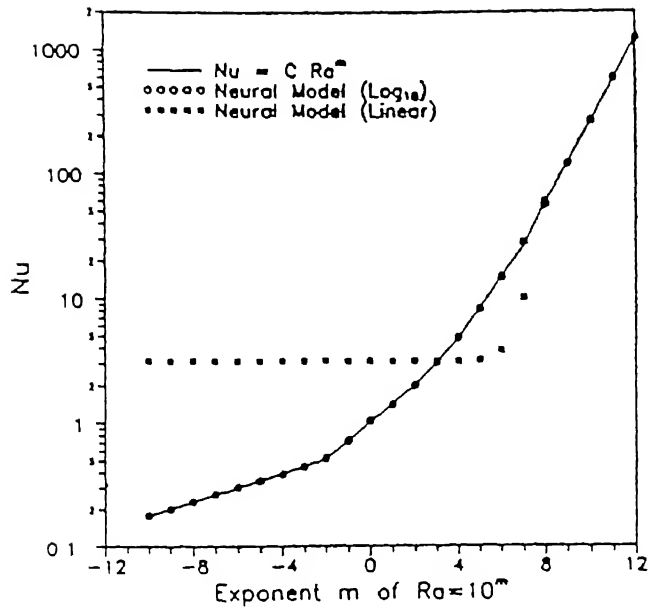
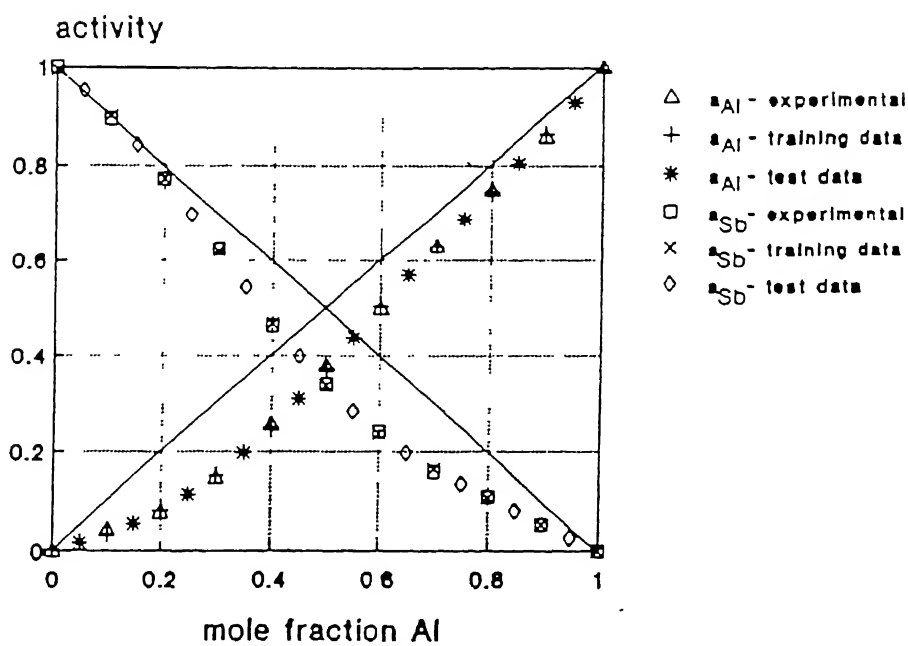


Figure 1.15 Nusselt number as function of Raleigh number for the free - convection around a smooth horizontal cylinder [7]



depicted with the help of two mathematical expressions. In addition to the three independent variables (Pr,n and Ω), the highly non-linear equations contain 32 parameters. So this example was chosen by the authors because of its involved complexity.

The architecture of the net contained 4 neurons (including bias) in the input layer and 2 neurons in the output layer (because local and average Nusselt numbers were the two dependent variables). The number of hidden neurons used were 6 in one case and 10 in the other. The results obtained showed that the performance of neural network was comparable, if not better, than that obtained by these model equations. The generalization ability of the net with 10 hidden neurons was better than that of with 6 hidden neurons. Another way of improving the function of the net could be by simplifying the structure using one output which means each net is used to predict only one type of Nusselt number.

1.2.3 Application of ANN in Slag-Metal Equilibrium

One of the important applications of ANN in process metallurgy is prediction of slag-metal equilibrium [8]. A number of theoretical models exist for describing the metal-slag equilibrium process in pyrometallurgy. Reuter *et. al.* used ANN to predict activities in binary metal-solution, like As-Sb. A regular- associated-solution (RAS) model was applied to generate the training data on activities of Al and Sb at 1400 K. The results obtained by ANN had an average error of 5.5% (Fig.1.16). One input node was the mole fraction of Al and two output nodes were the activities of Al and Sb. In all the layers, except the outermost one, a bias node was added. The input was scaled linearly and the outputs logarithmically. The ANN was trained with 11 data set and then tested upon 10 sets. The

prediction of ANN during training for ai_{Al} and a_{Sb} had error of 3.5% and 1.1% respectively. When error was evaluated with the test data for the same parameters, values of error 3.4% and 1.0% were obtained. It is thus clear that ANN predictions are superior to that obtained with regular associated solution (RAS) model. It is also evident from the Fig.(1.16) that the test data lie on the activity-concentration curves for the experimental data and it is possible to interpolate other data very well.

Reuter *et. al.* [8] also simulated equilibrium in iron-making. In the first example on the distribution of *Mn* between slag and metal, the two input variables were ($\%CaO/\%SiO_2$) and [Si] and output node was [%Mn]/(%Mn). The network had 48 numbers of hidden neurons. In all 48 data points were used for training and 7 data points for testing. The training error (after convergence) was found to be 5.6% and after testing the error value averaged 4.0%. The predicted results of [%Mn]/(%Mn)as obtained by ANN are shown in Fig.(1.17). It is clear that ANN was able to predict the seven test data points accurately.

In the second example ANN was used to predict sulphur distribution between slag and metal. Earlier workers investigated the sulphur distribution at 1500°C in an alumina crucible. Two types of experiments were performed in this crucible, the first in which sulphur was added to the metal and in the second case sulphur was added to the slag. The ANN was trained for both the cases and it had 4 input variables, namely ($\%CaO)/(\%SiO_2$), ($\%Al_2O_3$) , (%FeO) and [%S], 5 hidden nodes and 1 node for the output ($S)/[a_s]$. The net was trained with 9 data sets, while its performance was evaluated by 2 data points. The number of data points were not adequate to judge the generalization ability of an ANN. The results are shown in the Fig.(1.18). It can be clearly seen that in the second case the performance of the net is poorer than in the first case. This result was substantiated the observation that when sulphur was added to the slag equilibrium was not achieved.

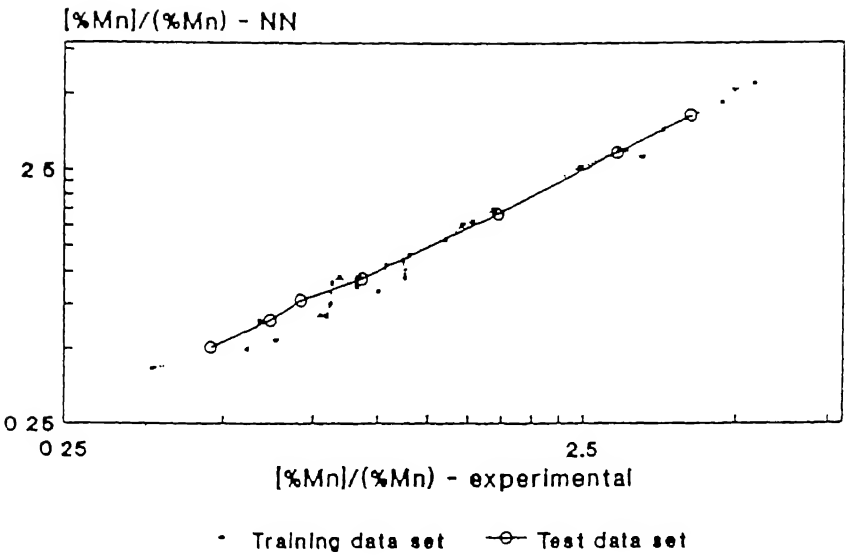


Figure 1 17 ANN predicted *vs* experimental values of $[\%Mn]/(\%Mn)$ - training and test data set [8]

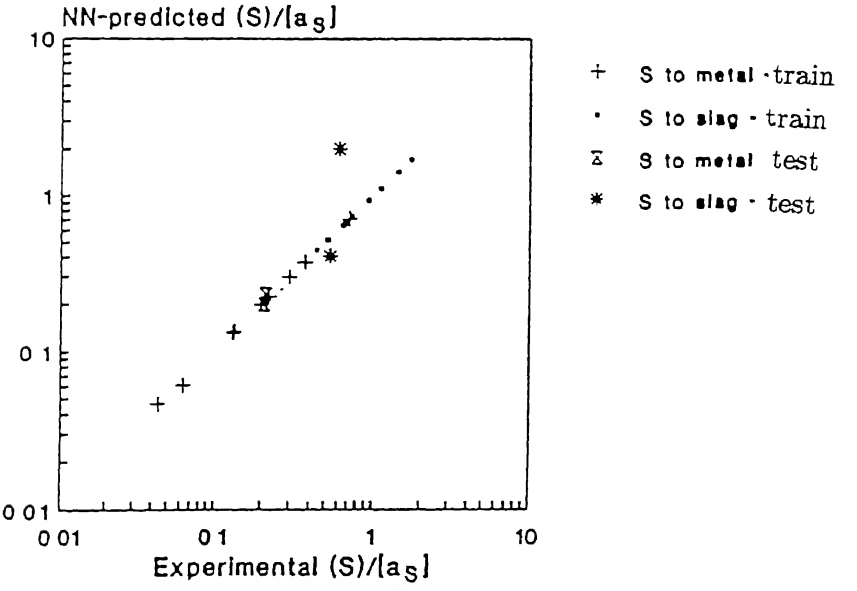


Figure 1.18 ANN predicted *vs* experimental values of $(S)/[a_S]$ for both S added to - metal and to slag - training and test data set [8]

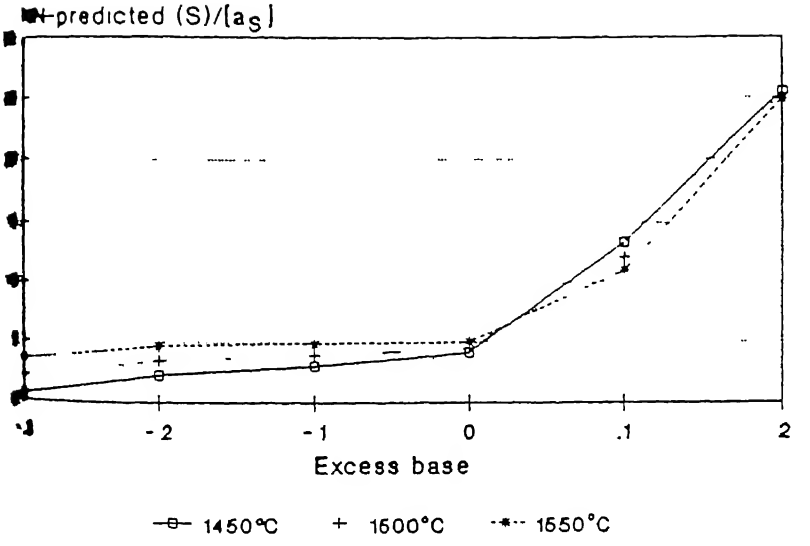


Figure 1.19 ANN prediction of $(S)/[a_S]$ as function of temperature and excess base for equilibrium experiments performed in a carbon crucible [8]

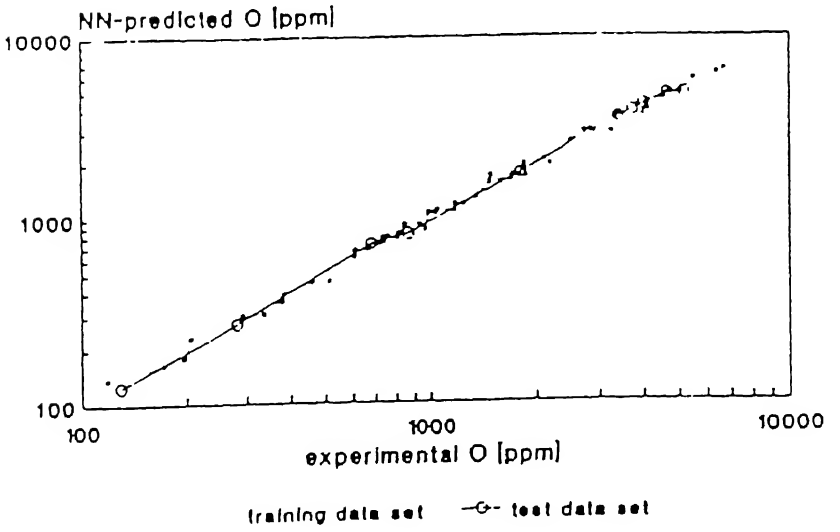


Figure 1 20 ANN predicted *vs* experimental values oxygen concentration - training and test data set [8]

Further experiments were conducted in graphite crucibles and sulphur distribution was investigated as a function of the excess base. An ANN with two input nodes, namely excess base and temperature (°C), three hidden nodes and one output node (for (S)/[a_s]), was trained on the experimental data. The training data set contained 17 patterns and error after training was 23.6% . The relatively large error during training was attributed by the authors to the noise in the original experimental data [3]. The results are shown in Fig.(1.19).

Both these examples of Mn and S distribution between slag and pig iron show how an ANN can be implemented to analyze experimental data, show trends and point out inconsistencies or inaccuracies in the experimental results.

In the third example oxygen concentration in copper melt as function of temperature was simulated. The data for the concentration of oxygen (ppm) in copper as a function of temperature was measured with the help of Ni-NiO and Co-CoO and a solid electrolyte reference electrodes. The ANN model developed was trained to map O content of copper melt (ppm) as a function of temperature (K) and emf(mV). The net had two inputs nodes, three hidden nodes and one output node. Sixty data points were employed for training, while generalization ability of the net was evaluated by ten data points. The average errors were 5.2% during training and 4.7% for testing. The results, as are shown in Fig.(1.20), highlight the consistent performance of the ANN in this particular applications as well.

In the fourth example, the ANN was used to predict distribution of copper between slag and metal. The data were taken from the works of earlier researchers [9] in which the solubility of copper was measured in a silica-saturated, iron-silicate slag in the temperature range of 1224°C - 1286°C. The partial pressure of oxygen ranged between $10^{-11.5}$ - $10^{-5.8}$ atm. The ANN model was used to simulate %Cu in slag as a function of process

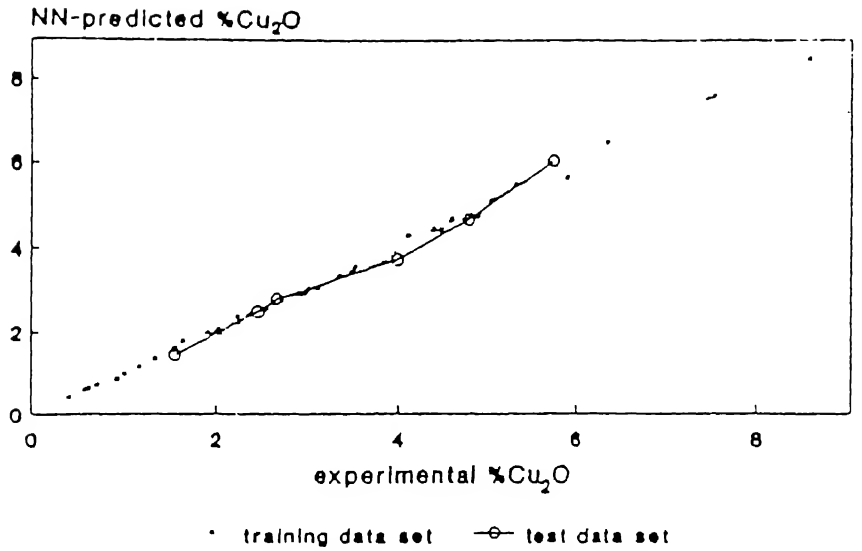


Figure 1.21 ANN predicted *vs* experimental values of $\%Cu_2O$ in slag - training and test data set [8]

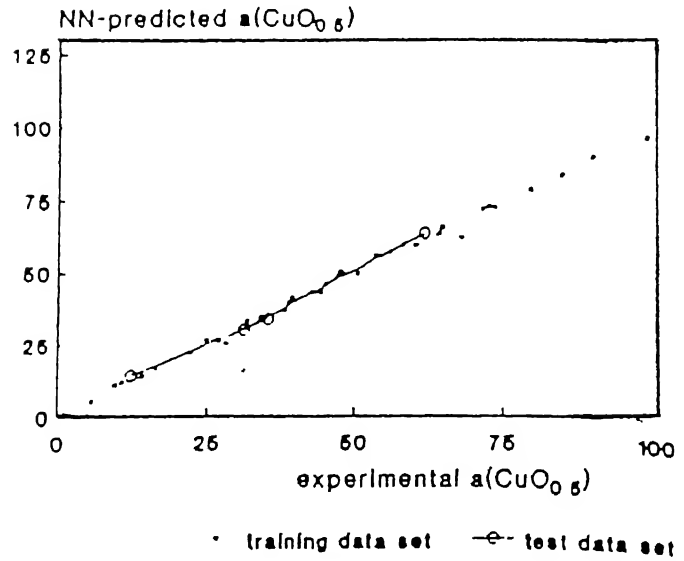


Figure 1.22 ANN predicted *vs* experimental values of $a(CuO_{0.5})$ - training and test data set [8]

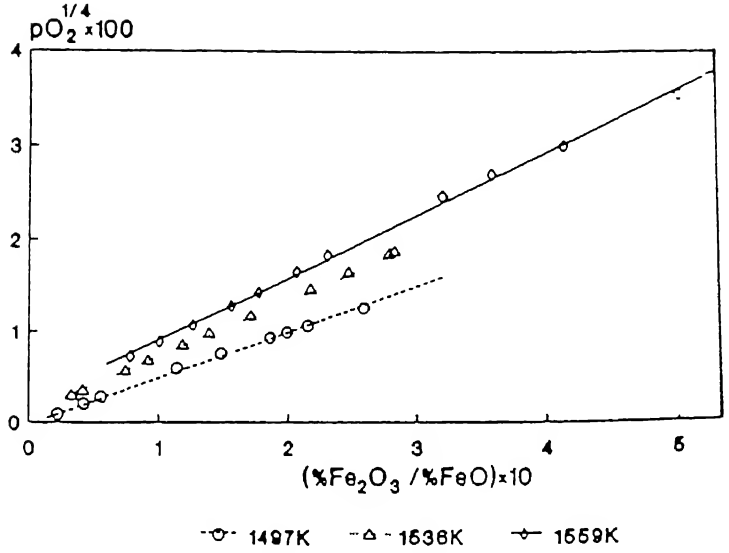


Figure 1 23 ANN predictions of $pO_2^{0.25}(\times 100)$ as function of $\%Fe_2O_3$ /$\%FeO(\times 100)$ for three given temperature [8]

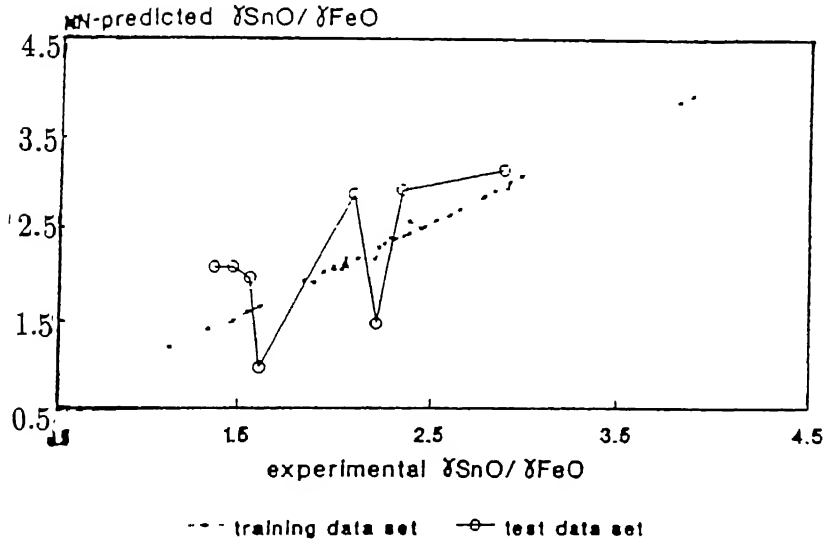


Figure 1 24 ANN predicted *vs* experimental values of $\gamma SnO/\gamma FeO$ with $\%Fe$ as one input - training and test data set [8]

conditions because Cu gets lost in slag in the form of Cu_2O. The Cu_2O content of slag was modeled as a function of four input variables, including temperature, pCO_2/pCO, $\%SiO_2$ and $\%FeO/\%Fe_2O_3$. The ANN architecture contained four input neurons, four hidden neurons and a single output neuron. Forty patterns were used for training and testing was done on six patterns. The error obtained in training was 2.2% and the error during testing was %4.9 (Fig.1.21). In order to determine the activity of $CuO_{0.5}$ in slag as a function of process conditions data were taken from previous work [9]. The ANN architecture consisted of three inputs (temperature, pCO_2/pCO, and %Cu) and one output ($a(CuO_{0.5})$) in slag. Four hidden neurons were found to be adequate. With 42 training data the average error was 3.0%, while for 4 training data the error was 6.2% (Fig.1.22).

A correlation was obtained by the authors between partial pressure of $O_2^{1/4}$ and the ratio of Fe_2O_3 and FeO content of slag. Temperature and the ratio of the oxides were used as two inputs and the only output was partial pressure of $O_2^{1/4}$. The number of hidden neuron were four. The results are shown in Fig.(1.24). One important observation of the study was that the data found out by Kellog *et. al.* [9] (which were used by the authors) followed slightly non-linear relationship, as expected from theory.

Two different ANN models were constructed to find out the dependence of the ratio of activity coefficients of SnO and FeO on the process conditions. Both of the models consisted of five hidden neurons and a single output for $\gamma SnO/\gamma FeO$. In the first model the four inputs were $(\%CaO)/(\%SiO_2)$, $(\%FeO)/(\%SiO_2)$, $\%Al_2O_3$ and %SnO, but in the second one extra neuron for %Fe in metal was added. In both the cases the error obtained during training was nearly same, but the error in testing was lower for the second net. The results are shown Fig.(1.24) and Fig.(1.25), respectively. The large scatter of the data during prediction in case of the first ANN model (Fig.1.24) may be attributed to the sig-

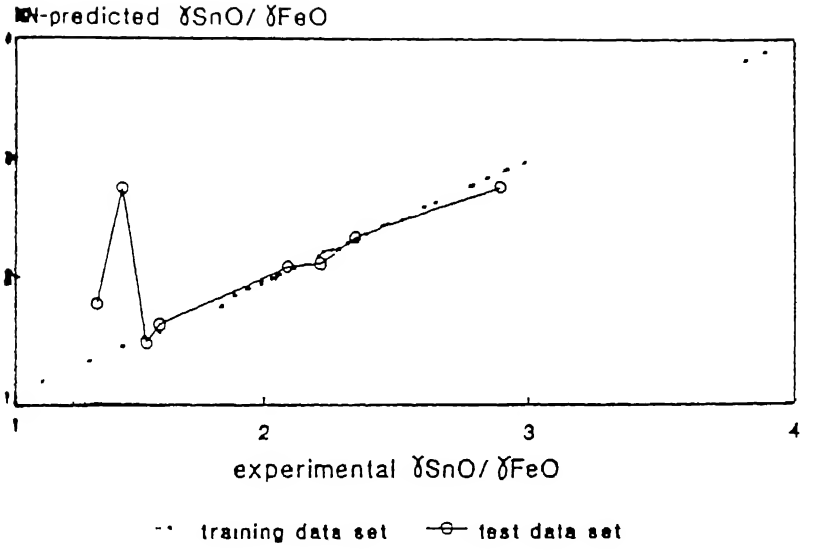


Figure 1 25 ANN predicted *vs* experimental values of $\gamma SnO/\gamma FeO$ without $\%Fe$ as 5 input - training and test data set [8]

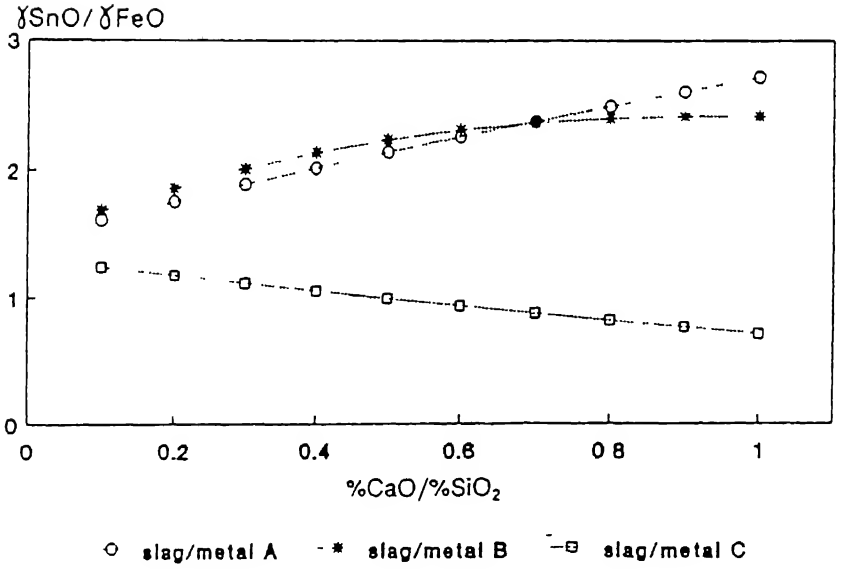


Figure 1.26 The effect of basicity on the $\gamma SnO/\gamma FeO$ ratio for three different slag - composition estimated by ANN [8]

nificant influence of Fe of metal on the $\gamma SnO/\gamma FeO$. It also shows when more number of significant parameters are included in the net architecture then the net can make better mapping and predictions.

In another study on Tin pyrometallurgy, relationship between the same ratio of activity coefficients with the basicity was established. A sensitivity analysis was performed using second ANN in the previous example. Three different types of slags were selected from experimental data for this analysis. They are slag/metal (A)-0.12% Fe (metal), 34.72% SnO and 1.62% Al_2O_3 with a $(\%FeO)/(\%SiO_2)$ value of 0.16, slag/metal (B)- 1.16%Fe (metal), 11.21% SnO and 2.25% Al_2O_3 with a $(\%FeO)/(\%SiO_2)$ of 4.43, slag/metal (C) - 2.54%Fe (metal), 8.21% SnO, 5.5% Al_2O_3 with value of 0.4 for $(\%FeO)/(\%SiO_2)$. The results are shown in Fig.(1.26). Results obtained by ANN are consistent with the experimental results reported by the Rankin [10].

In the last example a physical property like viscosity of Lead-Smelting slags was simulated. Previous workers correlated viscosity data with weight parameters by a fifth-order polynomial curve [11]. This technique is a tedious one because the workers could not fit the entire curve by a single equation and and a separate equations was needed for each temperature. The ANN approach is simpler and temperature was also incorporated in the structure. The ANN consisted of two inputs, namely weight parameters and temperature, three hidden neuron and one output for viscosity. After training and testing a relatively large error (around 33%) was found. But this value corresponded to that calculated by the earlier workers whose data were used in the present work. Still ANN had an edge over the other methods because temperature was included as one of the inputs. The results are shown in Fig.(1.27)

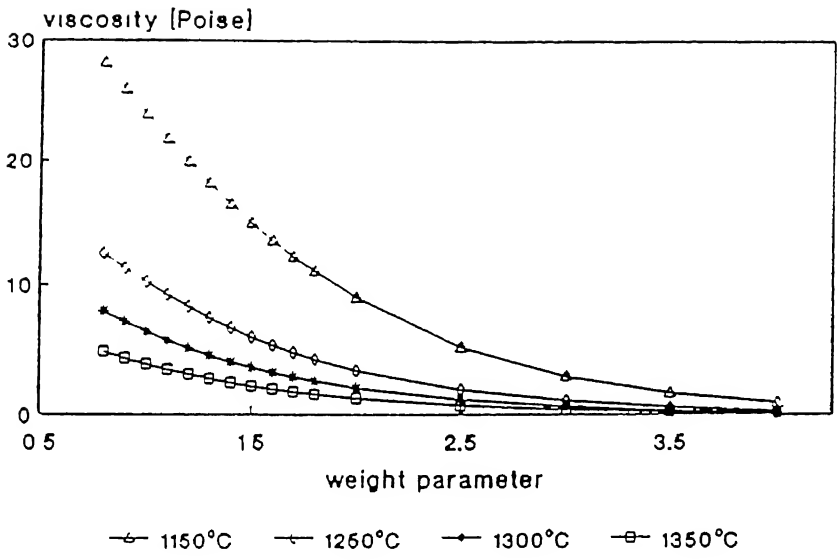


Figure 1.27 Viscosity as a function of the weight parameter and temperature, as predicted by the trained ANN [8]

1.2.4 Application of ANN in Electric Arc Furnace (EAF).

Effective control of electric arc furnace electrodes is a primary requirement for smooth furnace operation. Regulators, which drive an electrode positioning subsystem, enable many facets of furnace operation to be controlled. An effective regulator control results in control of secondary current, arc length, power factor, energy input rate (kW), furnace wall refractory wear and electrode usage. Staib *et. al.* [12] developed a control system based on ANN and showed that ANN mode was more time and cost effective than any other rule-based technique.

1.3 Scope of the Present Work

In the present work some significant aspects of ANN in the back-ground of desulphurization and dephosphorization operations of iron and steel making are discussed. The general training strategy along with the simulation results is described in chapter 2. Here the methodology to determine the net architecture including use of bias node and changes in net parameters (α, β and T) are explained. Some important and new modifications in ANN have been tried to improve the performance of the ANN and they are described in chapter 3. Optimization of ANN model has been done with the help of dynamic learning rate (β), pre-processing of supervising data and simplification of net configuration with the help of mathematical model optimized by Genetic Algorithm. Applicability and reliability of ANN models are discussed separately in chapter 4.

Chapter 2
ANN Simulation for Desulphurization and Dephosphorization in Iron and Steelmaking

In this chapter desulphurization hot metal and steel and dephosphorization of steel are simulated by Adaptive Neural Net (ANN). A brief description of these techniques are given below.

2.1 Desulphurization of Hot-metal in Torpedo

The interaction between the injected gas plus powder and the liquid metal is complex because of presence of several interfaces. The reactor model of the ladle injection is shown in Fig.(2.1). The powder particles are injected by high velocity carrier gas stream. The gas expands at the lance tip due to pressure drop and rise in temperature. If the velocity at the lance tip is sufficiently high then gas jet cone forms, otherwise bubbling takes place. In most of the metallurgical applications of this sort, jetting is likely to occur during powder injection.

The stream of particles coming out of the lance tip, initially travel in a straight line because of their high-momentum, but after sometime the gas jet breaks up into swarm of rising gas bubbles due to the resistance offered by metal bath and buoyancy forces. Some particles may penetrate into the liquid metal while the majority of them may become trapped inside the bubbles because of frictional and surface tension forces. The cone formed by the rising gas bubbles is known as 'gas plume'. If during their rise the particles are wetted by the liquid metal then they may break away from the bubble or be recirculated

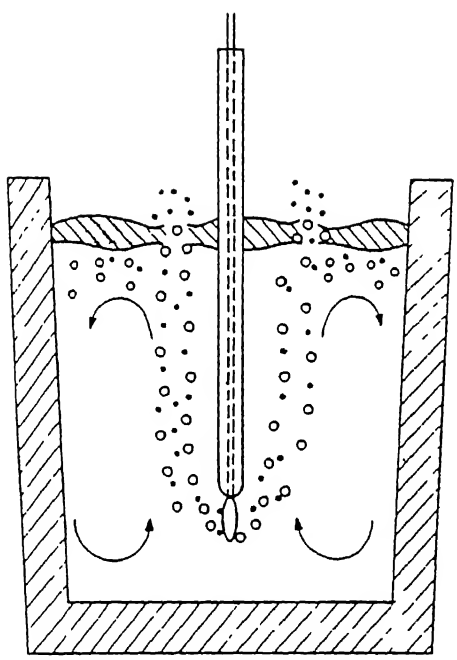


Figure 2.1 Reactor model for ladle injection [13]

back in the metal after reaching the top surface. Some particles may coagulate due to melting. A slag-metal-gas emulsion may form near the top surface and bigger bubbles may break into smaller ones also. A kinetic model of the desulphurization of liquid iron by the injection of calcium carbide has been developed by Deo and Boom [13], which is briefly summarized below.

The overall desulphurization rate during powder injection can be expressed as the sum of the rates of permanent and transitory reactions. By definition permanent reaction takes place due to top slag alone and transitory reaction takes place while the injected powder particles rise in the plume. In transitory mode various trajectories are possible for the powder particles. Some particles may sit inside the bubble, which are termed as particle inside the bubble (PIB), while others may rise freely and they are known as freely rising particle (FRP). All these have separate contribution towards desulphurization. The final model developed, which is applicable to desulphurization in a torpedo vessel is as follows.

$$ln\left(\frac{(\%S)_i}{(\%S)_f}\right) = (a+b+c)\frac{t_i n}{V} \qquad 2.1$$

where

$$\begin{aligned} a &= A_f k_f \\ b &= \frac{L.W(1-f)}{\rho_s}\left[1 - exp\left(\frac{-6k_p t_{rp}}{d_p L}\right)\right] \\ c &= \frac{L.W.f)}{\rho_s}\left[1 - exp\left(\frac{-2.38}{d_b} m \frac{T_{bath}}{298}\frac{k_g.t_{rb}.Q_{stp}.\rho_{sl}}{W.f.L}\right)\right] \end{aligned}$$

Here A_f is the nominal area of slag-metal contact (m^2), k_f is the mass transfer coefficient for permanent reaction (m/s), k_p is the mass transfer coefficient in metal around the FRP (m/s), k_g is the mass transfer coefficient of sulphur in metal for PIB (m/s), L is the

partition coefficient, f is the fractional of particles residing inside the bubbles, t_{rp} and t_{rb} are the residence time of particles in plume and of bubbles in bath in seconds, respectively, ρ_s and ρ_{sl} are the density of metal and slag (kg/m^3), respectively, d_p and d_b are the average diameter of particles and bubbles (m), respectively, T_{bath} is the temperature of bath (K), Q_{stp} is gas flow rate (m^3/s) at stp, m is the fraction of bubble base area taking part in desulphurization, t_{in} is the injection time (s), V is the volume of metal bath (m^3), and S_i and S_f are the initial and final sulphur content (wt%) of metal.

In the torpedo vessel, the desulphurization of hot-metal, is achieved by injection of powdered Calcium Diamide or CAD (CaC_2 + $CaCO_3$+ CaO) through a submerged lance into a torpedo, using nitrogen as carrier gas. The final sulphur content of hot-metal can be considered to be a function of five input variables, namely treatment time (s), hot-metal weight (ton), initial sulphur content (wt%), carrier gas flow rate (Nm^3/min.) and powder injection rate (kg/min). These five input variables constitute five nodes in the input layer of ANN and the output layer consists of one node (final sulphur content of hot-metal at the end of treatment). In order to decide the training strategy for a net and make the net conversant with as many patterns as possible, it was decided to train the net first with 16 patterns of input-output combination to save computer time and later on to increase number of patterns to 24 and then to 40 [Table-2.1]; one set of input-output data is called one pattern. All the patterns were scaled between 0.0 and 1.0 before introducing to the ANN. The initial weights were randomized between -0.5 to +0.5. The training strategy established by training nets with 16 patterns is described below and the same procedure was followed for training with 24 and 40 patterns. The first step was to decide a suitable net configuration (i.e. total number of layers and number of neurons in each layer). The various net architectures that were used to study the influence of number of layers and

Table 2.1 Training patterns for different net for desulphurization in 400 ton torpedo.

Serial No. -	Treatment Time (s)	Hot-metal weight (ton)	Initial wt %S	Gas Flow rate (Nm^3/min)	CAD Flow rate (kg/min)	Final wt %S
1	699	267	0.021	0.6	50.0	0.012
2	1044	271	0.036	0.6	55.0	0.014
3	652	307	0.021	0.6	57.0	0.007
4	904	289	0.030	0.6	55.0	0.015
5	961	340	0.033	0.6	57.0	0.016
6	1719	304	0.035	0.6	51.0	0.004
7	1446	335	0.029	0.6	54.0	0.016
8	1324	332	0.044	0.6	54.0	0.011
9	727	302	0.025	0.6	57.0	0.009
10	1363	362	0.044	0.6	54.0	0.013
11	650	329	0.020	0.6	70.0	0.010
12	942	300	0.034	0.6	62.0	0.011
13	1264	251	0.023	0.6	56.0	0.004
14	1164	379	0.023	0.6	48.0	0.005
15	651	268	0.009	0.7	45.0	0.009
16	791	293	0.027	0.6	54.0	0.006
17	687	264	0.027	0.6	65.0	0.008
18	984	280	0.020	0.6	59.0	0.003
19	1614	303	0.027	0.6	62.0	0.003
20	473	303	0.016	0.6	47.0	0.005
21	735	318	0.014	0.6	70.0	0.005
22	630	311	0.036	0.6	71.0	0.014
23	1642	317	0.030	0.6	64.0	0.003
24	920	335	0.039	0.7	67.0	0.018
25	814	351	0.018	0.6	48.0	0.009
26	1436	305	0.025	0.6	56.0	0.006
27	1003	316	0.017	0.6	55.0	0.003
28	988	339	0.042	0.6	66.0	0.011
29	1114	361	0.034	0.6	58.0	0.010
30	818	303	0.028	0.7	63.0	0.010
31	619	323	0.028	0.6	65.0	0.016
32	1337	319	0.037	0.6	46.0	0.010
33	1108	326	0.018	0.7	57.0	0.004
34	689	250	0.024	0.6	59.0	0.009
35	873	253	0.030	0.6	63.0	0.007
36	840	317	0.016	0.6	65.0	0.005
37	581	275	0.027	0.6	69.0	0.015
38	485	264	0.016	0.6	63.0	0.009
39	708	326	0.022	0.6	66.0	0.010
40	1003	255	0.039	0.6	59.0	0.007

number of neurons in each layer, are shown in Table-2.2. Bias neuron was not used in the ANN structure. To compare the learning behavior of these nets, learning rate(β), momentum factor(α) and logistics(T) were set at 0.5, 0.5 and 0.1, respectively. The criterion for judging the performance of the net with single output, is convergence of the error given by Eqn.(2.2). In case of net with more than one output, sum of errors of the individual outputs should converge during training. All the nets were trained over 10,000 iterations.

$$Error = \sqrt{\frac{\Sigma_{k=1}^{n}(o_k - t_k)^2}{n}} \qquad 2.2$$

where

o_k = predicted output for pattern k

t_k = target output for pattern k

n = number of training patterns

a) Effect of number of hidden neurons :

Fig.(2.2) shows the variation of error defined in Eqn.(2.2) with number of iterations for three-layered nets with different number of hidden neurons; the nets with fewer hidden neurons stop learning at high values of error. Also, beyond a certain optimum number of hidden neurons, the learning curves start showing perturbations as in the case of the nets [5,9,1] and [5,15,1]; these digits in bracket correspond to the neurons in input layer, hidden layer and output layer respectively. A [5,8,1] net configuration impl ies 5 neurons in the input layer, 8 neurons in the hidden layer and 1 neuron in the output layer.

Table 2.2 Net Configurations, trained for desulphurization in torpedo

Serial No.	No. of neurons				
	Input Layer	Hidden Layer-1	Hidden Layer-2	Hidden Layer-3	Output Layer
1	5	-	-	-	1
2	5	1	-	-	1
3	5	2	-	-	1
4	5	5	-	-	1
5	5	6	-	-	1
6	5	7	-	-	1
7	5	8	-	-	1
8	5	9	-	-	1
9	5	15	-	-	1
10	5	3	4	-	1
11	5	4	3	-	1
12	5	5	2	-	1
13	5	4	2	1	1

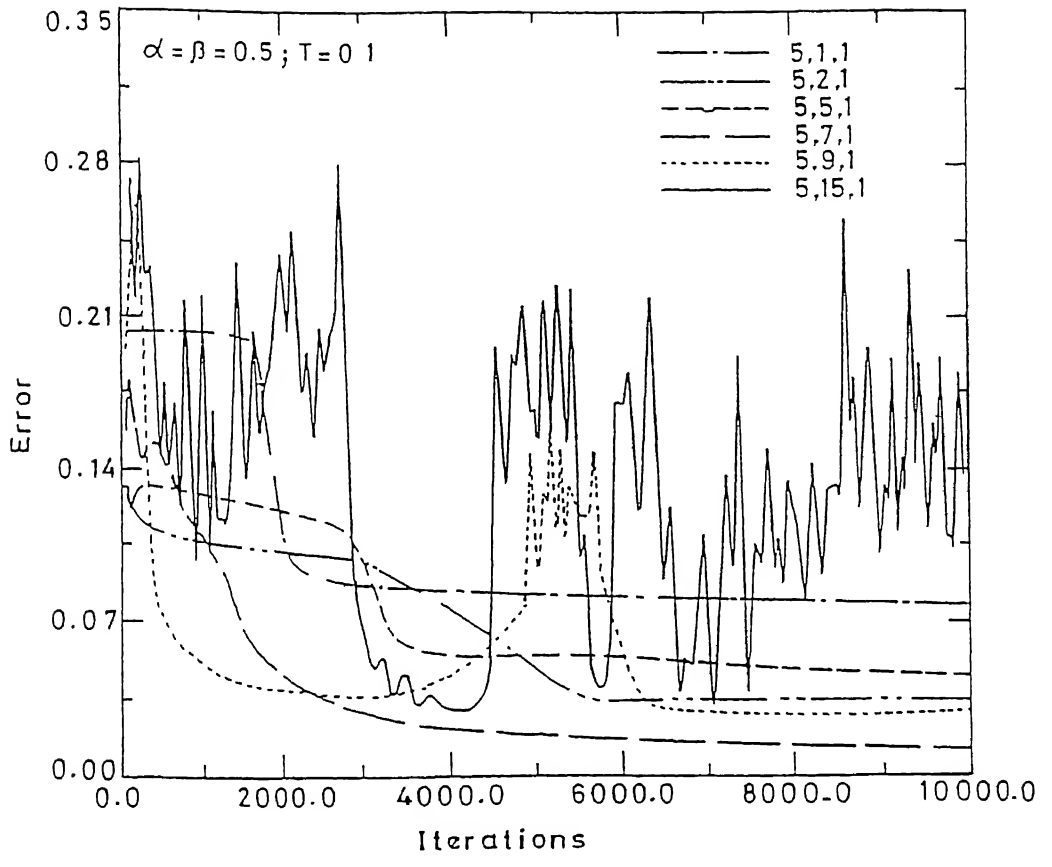


Figure 2 2 Effect of number of hidden neurons on learning by several net configurations for desulphurization of hot-metal in 400 ton torpedo.

b) Effect of distribution of hidden neurons :

Fig.(2.3) shows the effect on the learning curves when 7 hidden neurons are distributed in several combinations in hidden layers. It is observed that more layers do not necessarily improve the net performance (for given α, β and T values). For 16 patterns and earlier mentioned values of α, β, and T the net [5,7,1] is found to be an optimum one as it gives the lowest error at the end of 10,000 iterations; after 10000 iterations the error decreased very slowly.

c) Effect of changing logistics (T) :

Once a suitable net configuration has been chosen using the above mentioned criteria, the other net parameters may be varied to decrease error during training. It is advisable to choose the logistics (T) first, since the sigmoidal function in Eqn.(1.11) decides the "firing strength" of each neuron of the net. The results of the experiments conducted to determine the effects of T are shown in fig.(2.4). It is found that a value of T=0.01 (i.e. when the sigmoidal function approximates to a step function) is not suitable when used with [5,7,1] net (with α=β=0.5). Also, T=10 was found to be inappropriate because the learning curve flattens out (i.e. learning stops) at a relatively high error value, quite early during the training period. It is suggested to choose T, which exhibits lowest error. It was decided to keep the value of logistics(T) constant at 0.1 for further experiments.

d) Effects of changing momentum factor(α) and learning rate(β)

After choosing the net configuration and logistics(T), suitable values of α and β should be chosen. It can be seen that at high α value (say α=0.9) the learning process stops at very high error values, whereas at lower α (say α=0.5) the convergence is more rapid

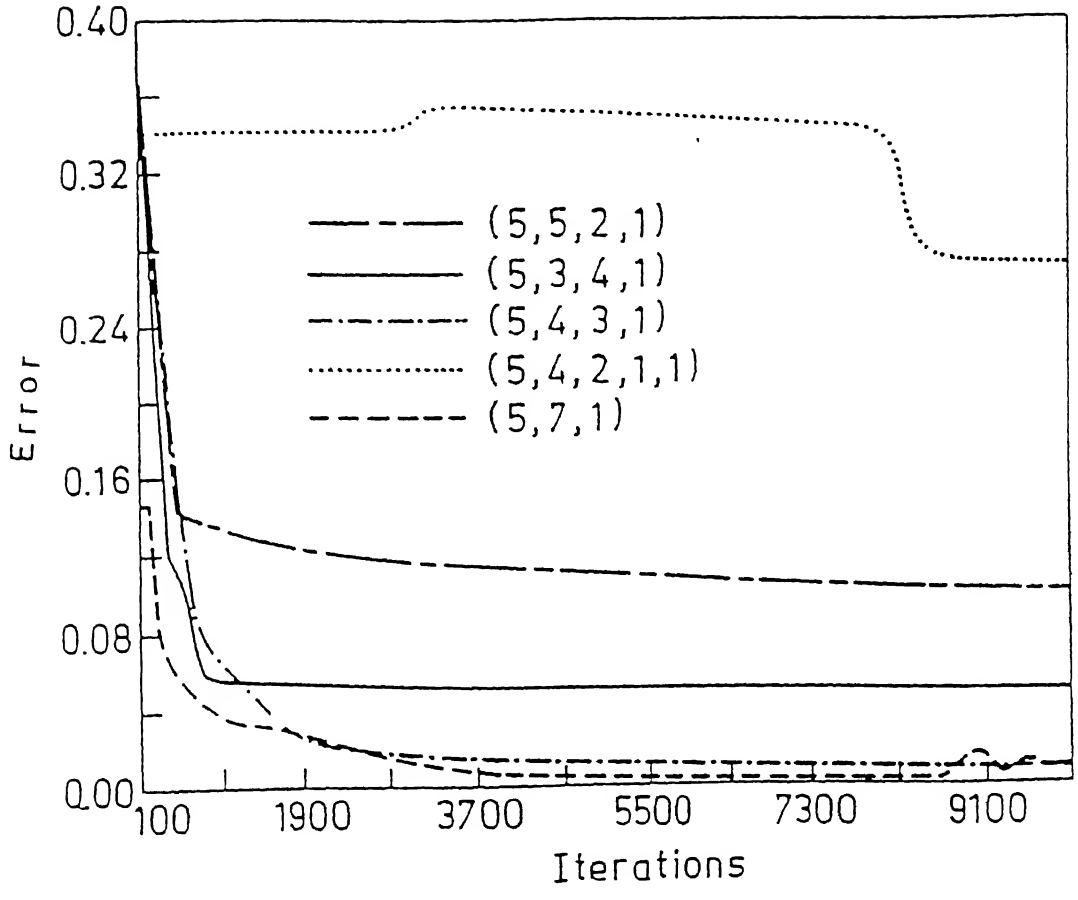


Figure 2.3 Effect of distribution of hidden neurons on learning by [5,7,1] net for desulphurization of hot-metal in 400 ton torpedo.

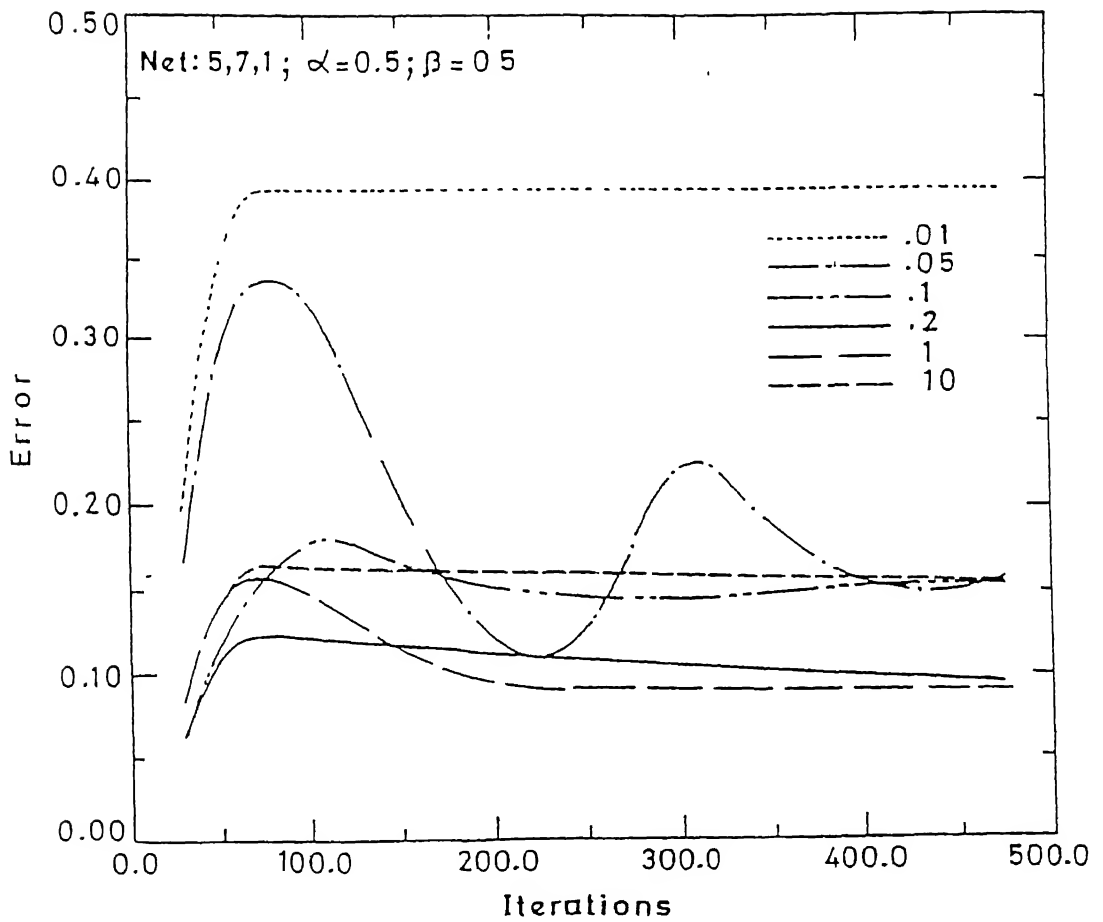


Figure 2 4 Effect of changing logistics on learning by [5,7,1] net for desulphurization of hot-metal in 400 ton torpedo.

[Fig.(2.5)]. It is found that higher learning rate (β=0.9) increases error and lower learning rate (β=0.5) gives smoothness to the learning curve as shown in Fig.(2.6). One interesting observation is that a low β smoothens out the perturbations caused by using higher number of hidden neuron, as illustrated in Fig.(2.7). Therefore, whenever the complexity of a problem demands for larger number of hidden neurons in the net, lower β may tend to improve learning characteristics.

e) Memorization performance of the net :

For illustration purposes, a [5,7,1] net with α=β=0.5 and T=0.1 was trained with 16 set of data, over 10,000 iterations. A comparison of the net's predictions for the same 16 sets and the actual target outputs is shown in Fig.(2.8). It is observed that the net has memorized the 16 sets remarkably well, which means the modification of interconnect weights proceeded in the right direction.

f) Effect of number of patterns for training :

From the study of 16 patterns it became evident that the optimum values of momentum factor (α) and learning rate (β) are 0.5 and logistics (T) is 0.1. Further training was done with 24 and 40 patterns. Training with 40 patterns required at least 15,000 iterations to converge. Test set patterns and test results with different optimum nets are given in Table-2.3. Number of training patterns, corresponding optimum net configuration along with the values of standard deviation (σ) and correlation coefficient (R) of the test results are tabulated in Table-2.4. Actual and predicted values of final sulphur content of the melt are also plotted in Fig.(2.9) for all these optimum net architectures in different cases. It is clear that as we make the net conversant with more number of training patterns its ability

Table 2.3 Patterns used for testing different nets and test results for desulphurization in 400 ton torpedo.

S. No	Time (s)	Metal (ton)	Initial wt%S	Gas flow rate (Nm^3 /min)	CAD flow rate (kg /min)	Actual final wt%S	Predicted final wt%S with increasing number of training patterns (16-24-40)		
							(16) [5,8,1]	(24) [5,8,1]	(40) [5,9,1]
1	1404	280	0.026	0.6	67.0	0.003	0.1436	0.0059	0.0030
2	1421	309	0.024	0.6	70.0	0.003	0.0150	0.0031	0.0030
3	473	303	0.016	0.6	62.0	0.005	0.0074	0.0033	0.0084
4	1161	362	0.030	0.6	73.0	0.005	0.0157	0.0030	0.0030
5	1085	336	0.028	0.6	63.0	0.008	0.0143	0.0090	0.0050
6	559	308	0.021	0.6	63.0	0.009	0.0090	0.0036	0.0149
7	483	304	0.018	0.6	58.0	0.011	0.0087	0.0043	0.0030
8	501	317	0.020	0.6	63.0	0.011	0.0091	0.0038	0.0096
9	779	295	0.030	0.6	68.0	0.012	0.0159	0.0161	0.0163
10	796	344	0.033	0.6	60.0	0.015	0.0160	0.0161	0.0163
11	455	271	0.022	0.6	68.0	0.017	0.0151	0.0042	0.0166

Table 2.4 Optimum net configurations (iterations : 15,000) and test results for desulphurization in 400 ton torpedo.

Training Patterns	Optimum Net Configuration	σ	R
16	[5,8,1]	0.0064	0.14
24	[5,8,1]	0.0038	0.46
40	[5,9,1]	0.0028	0.86

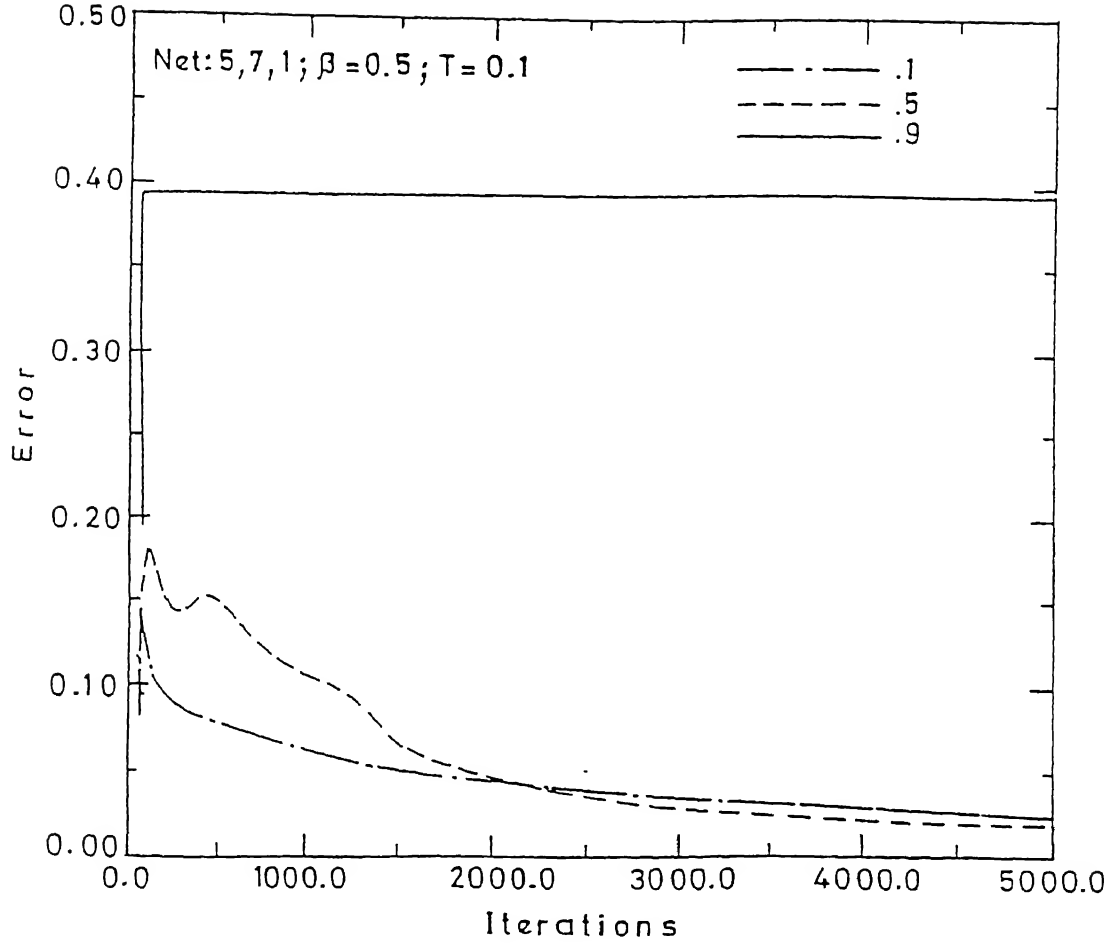


Figure 2.5 Effect of changing momentum factor on learning by [5,7,1] net for desulphurization of hot-metal in 400 ton torpedo.

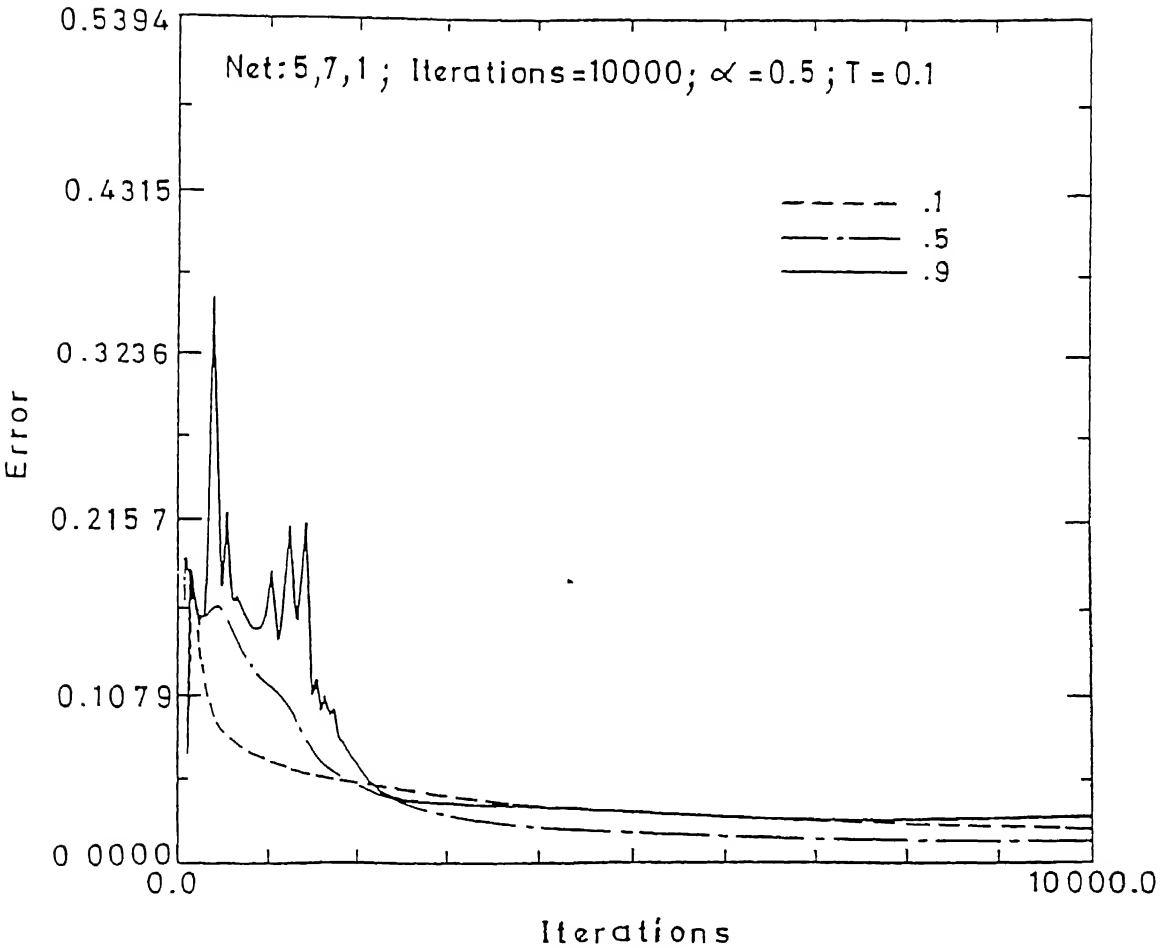


Figure 2.6 Effect of changing learning rate on learning of [5,7,1] net for desulphurization of hot-metal in 400 ton torpedo.

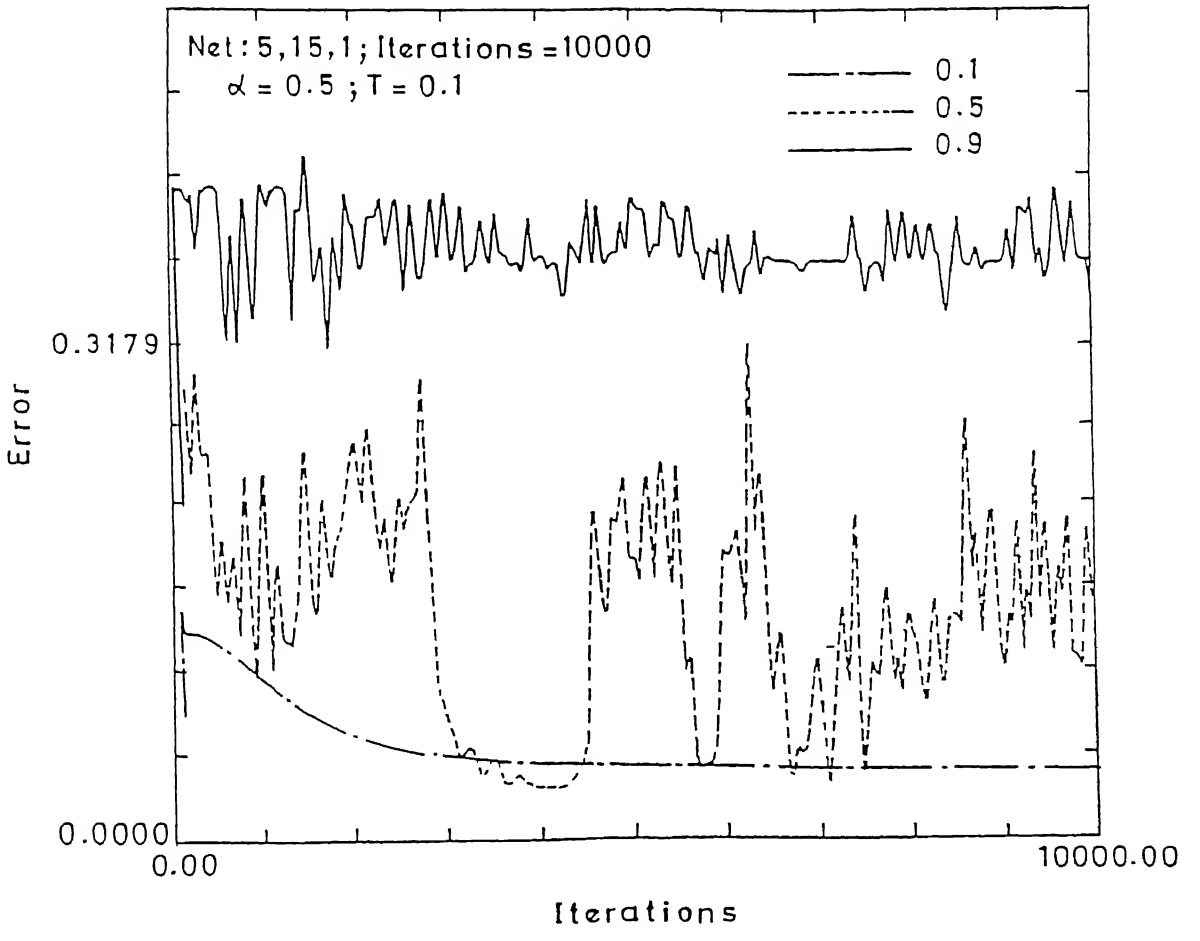


Figure 2.7 Effect of changing learning rate on learning of [5,15,1] net for desulphurization of hot-metal in 400 ton torpedo.

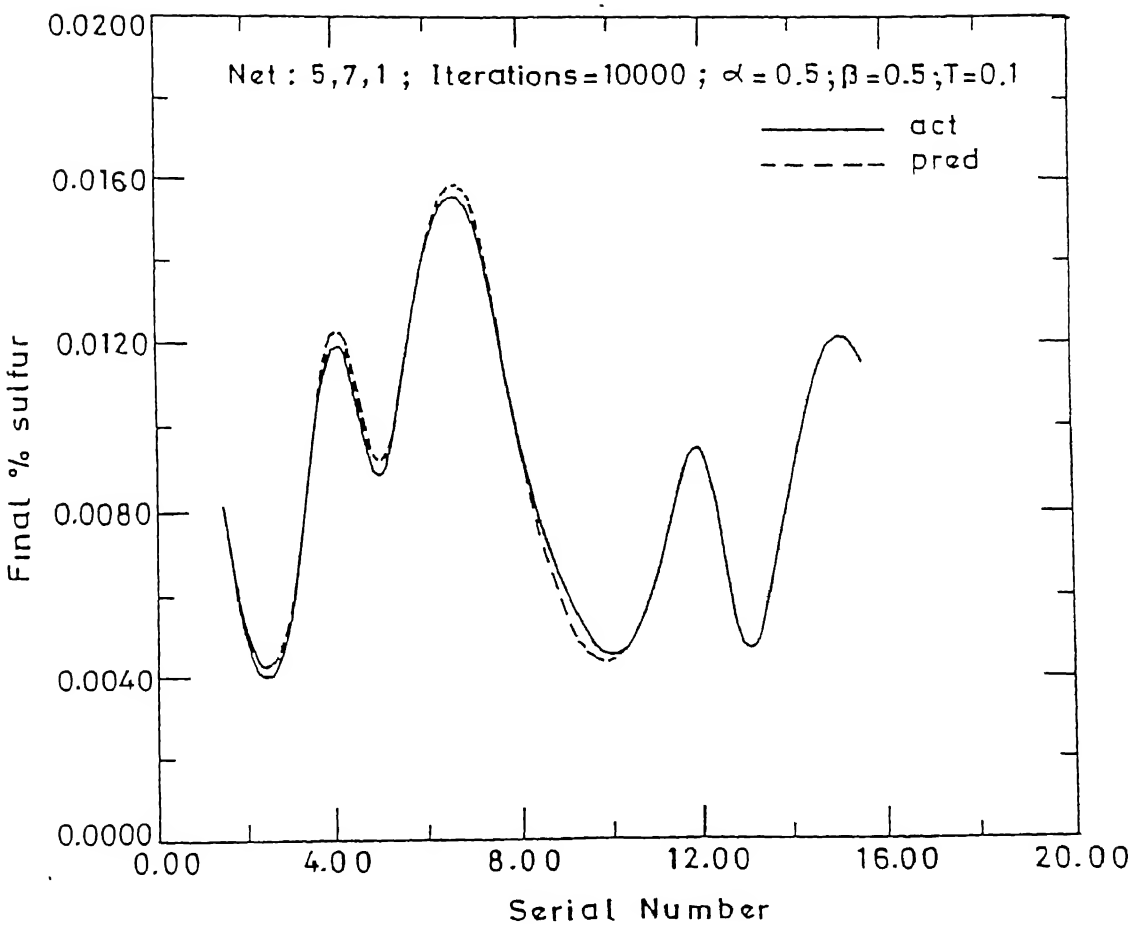


Figure 2.8 Memorization performance of [5,7,1] net for desulphurization of hot-metal in 400 ton torpedo.

Table 2.5 Training patterns for desulphurization of hot-metal in 400 ton torpedo.

Serial No. -	Initial wt %S	Hot-metal weight (ton)	CAD added (kg.)	Treatment time (min)	Final wt %S
1	0.038	342	880	17.60	0.010
2	0.036	314	930	19.79	0.011
3	0.032	342	1180	19.67	0.004
4	0.028	366	590	14.75	0.013
5	0.064	367	2480	57.67	0.003
6	0.035	339	1210	30.25	0.006
7	0.028	353	620	15.12	0.013
8	0.036	368	830	16.27	0.013
9	0.034	373	1220	32.97	0.006
10	0.023	318	890	21.19	0.005
11	0.036	353	1780	36.33	0.004
12	0.058	363	1330	31.67	0.009
13	0.046	358	1460	34.76	0.006
14	0.039	319	1000	23.81	0.012
15	0.040	310	850	22.97	0.014
16	0.051	328	1336	33.40	0.006
17	0.024	321	440	10.73	0.011
18	0.032	343	1310	28.48	0.004
19	0.043	353	1125	28.85	0.012
20	0.037	360	990	20.63	0.014
21	0.033	334	1140	25.91	0.004
22	0.066	356	1820	43.33	0.012
23	0.060	303	1750	41.67	0.004
24	0.039	334	810	27.00	0.015
25	0.064	348	1743	37.09	0.009
26	0.050	300	2060	47.91	0.003
27	0.035	306	1130	31.39	0.004
28	0.034	313	1450	32.95	0.002
29	0.030	299	1010	20.61	0.004
30	0.032	304	758	16.48	0.016
31	0.026	317	820	18.64	0.006
32	0.027	317	520	12.38	0.012
33	0.056	320	2400	61.54	0.002
34	0.042	296	1705	33.43	0.014
35	0.028	310	670	17.63	0.015
36	0.045	288	880	17.25	0.015
37	0.040	309	800	16.00	0.015
38	0.028	384	2960	67.27	0.001
39	0.039	325	1290	29.32	0.006
40	0.049	322	1100	22.92	0.008

continued...

Table 2.5 (continued..) Training patterns for desulphurization of hot-metal in 400 ton torpedo.

Serial No. -	Initial wt %S	Hot-metal weight (ton)	CAD added (kg.)	Treatment time (min)	Final wt %S
41	0.025	336	940	22.38	0.007
42	0.030	380	1090	30.28	0.004
43	0.036	347	870	17.40	0.010
44	0.050	408	2100	51.22	0.002
45	0.049	365	1610	36.59	0.006
46	0.047	357	1960	39.20	0.004
47	0.059	313	2100	42.00	0.009
48	0.058	376	1850	46.25	0.009
49	0.040	346	2030	42.29	0.002
50	0.049	386	2210	46.04	0.002
51	0.021	353	400	9.09	0.013
52	0.031	362	1100	26.19	0.006
53	0.038	376	1940	44.09	0.002
54	0.031	375	1610	28.75	0.003
55	0.027	385	1290	26.33	0.008
56	0.028	351	1120	31.11	0.005
57	0.058	347	2050	37.96	0.012
58	0.038	337	900	21.43	0.015
59	0.050	377	1870	36.67	0.003
60	0.059	353	1850	38.54	0.009
61	0.050	365	1210	27.50	0.013
62	0.042	340	1980	42.13	0.006
63	0.041	327	950	24.36	0.013
64	0.024	300	430	8.60	0.011
65	0.027	344	670	18.61	0.006
66	0.030	316	1150	25.00	0.004
67	0.034	301	830	19.30	0.008
68	0.036	316	1410	40.29	0.005
69	0.033	317	805	24.39	0.008
70	0.034	335	730	20.28	0.007
71	0.046	338	1330	32.44	0.007
72	0.036	310	770	15.40	0.015
73	0.040	289	820	19.52	0.010
74	0.030	306	1390	31.59	0.004
75	0.038	309	820	21.58	0.010
76	0.053	336	3160	65.83	0.005
77	0.021	294	370	8.22	0.011
78	0.028	338	1050	27.63	0.005
79	0.024	309	490	11.40	0.008
80	0.042	328	1455	31.63	0.005

continued

Table 2.5 (continued) Training patterns for desulphurization of hot-metal in 400 ton torpedo.

Serial No. -	Initial wt %S	Hot-metal weight (ton)	CAD added (kg.)	Treatment time (min)	Final wt %S
81	0.053	324	1695	42.38	0.008
82	0.041	339	1280	30.48	0.010
83	0.031	350	1546	34.36	0.004
84	0.040	324	2090	40.19	0.004
85	0.030	304	580	15.26	0.007
86	0.027	342	360	7.06	0.005
87	0.022	371	770	17.50	0.005
88	0.035	338	2040	39.23	0.002
89	0.048	324	2205	50.11	0.005
90	0.034	311	740	16.44	0.011
91	0.039	353	1390	40.88	0.005
92	0.043	332	330	8.46	0.008
93	0.054	313	1520	27.64	0.007
94	0.046	311	2500	54.35	0.004
95	0.039	280	330	7.86	0.009
96	0.025	331	535	12.74	0.012
97	0.032	337	1050	24.42	0.005
98	0.058	308	2510	46.48	0.006
99	0.030	331	540	10.59	0.010
100	0.033	342	1760	44.00	0.004
101	0.036	337	1685	27.62	0.005
102	0.078	356	1905	50.13	0.013
103	0.041	362	1870	42.50	0.003
104	0.057	355	2140	47.56	0.003
105	0.062	349	2260	56.50	0.003
106	0.053	351	2280	44.71	0.003
107	0.044	350	1400	35.90	0.010
108	0.029	322	7670	13.14	0.011
109	0.026	297	1260	28.00	0.003
110	0.028	309	940	18.43	0.008
111	0.047	317	940	27.65	0.015
112	0.040	310	1700	42.50	0.012
113	0.044	288	840	19.53	0.007
114	0.027	293	1190	36.06	0.002
115	0.025	350	500	11.36	0.012
116	0.023	350	790	18.81	0.005
117	0.035	335	1020	24.29	0.011
118	0.027	326	990	18.33	0.005
119	0.032	331	1150	28.05	0.004
120	0.032	317	750	27.78	0.005

to predict and generalize improves (σ decreases from 0.0064 to 0.0028 and R increases from 0.14 to 0.86 as training patterns are increased from 16 to 40).

g) Effect of bias node in net configuration :

It has been discussed earlier that input and output of any neuron is related by sigmoidal function given by Eqn.(1.11). The sigmoidal curve is asymptotic at both the ends. So input in the infinite range is required to achieve an output of 0 or 1. In order to tackle this problem a bias node is added in each layer of neuron except in the output layer (Fig.1.12). This bias or auxiliary node has an output of unity and it does not have any connection with the neurons of the previous layer. The weights between the bias and other nodes are trained like other connection weights only. Thus bias node shifts the position of the sigmoidal curve and tries to minimize the gap between actual and predicted output.

The influence of bias node is demonstrated in the case of hot-metal desulphurization in a 400 ton torpedo vessel. Only those heats are considered in which the carrier gas flow rate was constant at 2.0 m^3/min. The number of neurons in input layer are five including one bias node. The training set is given in Table 2.5. The optimum net configuration without bias node was found to be (4,9,1), which means four input nodes, nine hidden nodes and a single output. Later on bias was added in the same net structure. In order to take care of the asymptotic nature of the sigmoidal curve it was decided to scale the outputs between 0.1 to 0.9 along with the introduction of bias node. From Fig.(2.10) it can be easily inferred that introduction of bias node helps lowering the error after training.

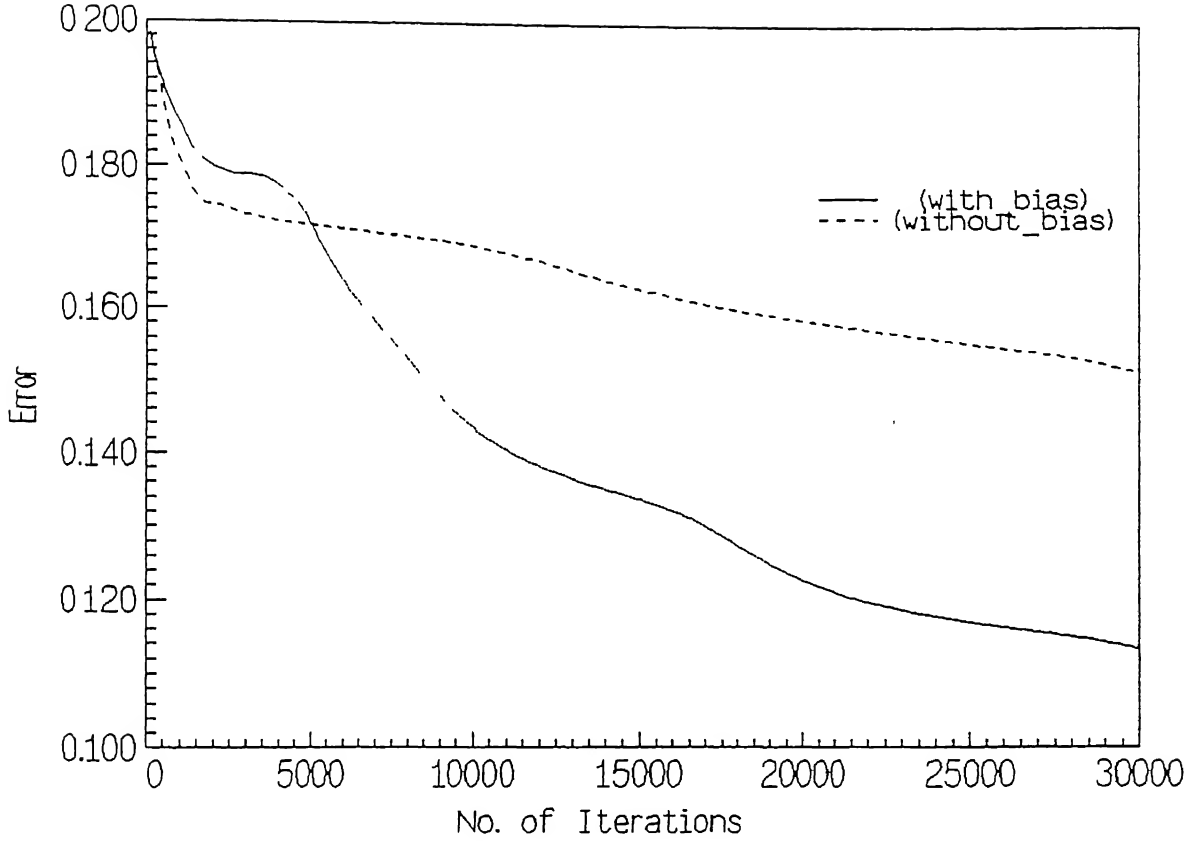


Figure 2.10 Effect of bias node on learning of [4,9,1] net for desulphurization of hot metal in 400 ton torpedo.

2.2 Desulphurization and Dephosphorization of Steel in Combined Blown Converter

In the next study desulphurization and dephosphorization in a 300 ton oxygen steelmaking combined blown converter are simulated by ANN. Oxygen steelmaking is a complex process and desulphurization at tap is affected by many parameters. Eight variables, namely metal weight, total amount of oxygen blown, amount of iron ore added, and temperature, contents of carbon , manganese, phosphorus, sulphur as determined by in-blow sampling, are considered as inputs to ANN while sulphur content of liquid steel at tap is the output variable. The strategy of training and testing of the net is similar to that of desulphurization of hot-metal, already discussed section 2.1. The nets are trained with 50 patterns [Table-2.6] and tested on another 50 patterns of data [Table-2.7]. The optimum net configuration is found to be (8,16,1). The results of testing are shown in Fig.(2.11). The value of standard deviation of the best-fit straight line passing through origin, considering actual and predicted values of final sulphur content of the melt as two variables, is 0.0019 and the correlation coefficient is 0.87.

The amount of phosphorus present in the steel after the operation is also considered to be a function of those eight variables mentioned above. The optimum net was found to be (8,16,1) when trained with 50 patterns [Table-2.8]. The generalization performance of the trained net was evaluated by another 50 set [Table-2.9]. Comparison of actual and predicted value of phosphorus content of the melt is depicted in Fig.(2.12). In this case the R and σ are 0.78 and 0.002, respectively.

It is worth mentioning that, due to the paucity of information the amount of lime and

Table 2.6 Training patterns for desulphurization of steel in 300 ton converter

S No -	Metal wt. (ton)	Ini--tial temp. (°C)	Ini--tial wt%C	Ini--tial wt%Mn	Ini--tial wt%P	Ini--tial wt%S	Ore added (Kg.)	Total O_2 blown (m^3)	Final wt%S
1	326.5	1620	0.541	0.354	0.034	0.014	1000	14310	0.012
2	315.0	1612	0.503	0.321	0.032	0.016	988	14390	0.013
3	310.2	1639	0.559	0.368	0.036	0.020	1004	14550	0.014
4	325.5	1608	0.484	0.314	0.024	0.018	1004	14370	0.014
5	315.1	1594	0.448	0.336	0.023	0.018	1008	13520	0.015
6	314.6	1606	0.420	0.338	0.031	0.020	500	13580	0.017
7	315.4	1607	0.525	0.338	0.037	0.028	1024	13870	0.019
8	313.5	1628	0.391	0.343	0.031	0.021	1010	13520	0.017
9	316.7	1629	0.485	0.334	0.037	0.017	1000	13920	0.013
10	314.6	1612	0.537	0.342	0.026	0.017	1500	13630	0.013
11	313.2	1622	0.405	0.358	0.033	0.017	998	14280	0.013
12	315.6	1623	0.463	0.374	0.036	0.019	1000	14190	0.015
13	315.4	1632	0.416	0.344	0.033	0.013	1000	14560	0.012
14	313.7	1657	0.130	0.264	0.015	0.012	598	13690	0.011
15	313.9	1642	0.406	0.372	0.033	0.015	1004	13990	0.013
16	316.1	1652	0.443	0.338	0.031	0.010	998	14020	0.008
17	315.7	1614	0.531	0.347	0.024	0.014	508	13810	0.011
18	317.0	1593	0.701	0.319	0.024	0.017	788	13980	0.014
19	316.5	1591	0.606	0.333	0.032	0.020	500	14220	0.015
20	315.6	1614	0.586	0.341	0.029	0.020	1000	14210	0.016
21	315.8	1616	0.568	0.341	0.029	0.020	1000	14210	0.016
22	315.2	1596	0.654	0.334	0.025	0.019	1008	14090	0.015
23	315.0	1603	0.323	0.252	0.014	0.020	500	13710	0.015
24	314.7	1618	0.515	0.340	0.027	0.015	1002	14100	0.013
25	310.1	1593	0.464	0.300	0.020	0.016	976	13980	0.014
26	313.4	1589	0.506	0.288	0.021	0.016	988	14050	0.014
27	314.0	1592	0.397	0.302	0.021	0.021	1000	14030	0.016
28	313.0	1595	0.291	0.303	0.020	0.018	1000	13820	0.013
29	313.7	1584	0.456	0.297	0.022	0.018	814	14250	0.014
30	314.2	1586	0.433	0.260	0.022	0.018	800	13970	0.016
31	315.6	1623	0.387	0.332	0.028	0.016	1004	14000	0.013
32	315.3	1617	0.357	0.324	0.028	0.015	998	13960	0.014
33	313.9	1632	0.232	0.310	0.026	0.020	498	14000	0.016
34	310.6	1641	0.497	0.348	0.039	0.019	1500	14010	0.014
35	314.7	1615	0.590	0.334	0.034	0.013	1010	14490	0.010

continued..

Table 2.6 (..continued) Training patterns for desulphurization of steel in 300 ton converter

S No -	Metal wt. (ton)	Ini- -tial temp. (°C)	Ini- -tial wt%C	Ini- -tial wt%Mn	Ini- -tial wt%P	Ini- -tial wt%S	Ore added (Kg.)	Total O_2 blown (m^3)	Final wt%S
36	315.5	1603	0.463	0.315	0.029	0.016	1000	13610	0.013
37	314.7	1603	0.380	0.260	0.016	0.037	1014	13820	0.015
38	314.6	1611	0.252	0.317	0.019	0.019	1000	13790	0.017
39	313.9	1625	0.299	0.320	0.027	0.017	1014	13680	0.014
40	314.5	1630	0.407	0.322	0.036	0.017	1000	14080	0.013
41	322.5	1633	0.318	0.315	0.039	0.016	1500	13890	0.011
42	316.0	1610	0.308	0.323	0.026	0.009	998	13480	0.008
43	315.2	1610	0.471	0.357	0.034	0.013	1004	13860	0.011
44	315.7	1613	0.313	0.315	0.026	0.019	998	13670	0.016
45	312.9	1629	0.447	0.345	0.032	0.017	1010	14440	0.015
46	314.4	1641	0.442	0.333	0.036	0.025	1002	13780	0.018
47	315.2	1625	0.326	0.329	0.023	0.018	1010	12970	0.016
48	315.0	1614	0.316	0.321	0.023	0.007	508	13880	0.006
49	314.2	1569	0.646	0.336	0.030	0.012	510	14490	0.009
50	305.3	1623	0.446	0.405	0.034	0.017	1000	13230	0.014

Table 2.7 Test set data and predictions for desulphurization of steel in 300 ton converter

S No -	Metal wt. (ton)	Ini--tial temp. (°C)	Ini--tial wt%C	Ini--tial wt%Mn	Ini--tial wt%P	Ini--tial wt%S	Ore added (Kg.)	Total O_2 blown (m^3)	Actual final wt%S	Predicted final wt%S
1	315.7	1604	0.374	0.329	0.022	0.010	514	13820	0.007	0.00612
2	314.4	1618	0.425	0.337	0.025	0.008	1000	14110	0.007	0.00693
3	314.1	1610	0.485	0.346	0.029	0.009	498	14530	0.007	0.00606
4	314.2	1603	0.512	0.337	0.030	0.008	790	13480	0.008	0.01112
5	314.1	1627	0.393	0.346	0.033	0.010	1000	13450	0.009	0.00681
6	314.9	1611	0.423	0.325	0.032	0.012	1014	13690	0.009	0.00894
7	312.9	1624	0.568	0.317	0.027	0.012	1000	13100	0.009	0.01792
8	313.1	1624	0.633	0.341	0.033	0.012	1004	14260	0.009	0.00822
9	313.3	1583	0.466	0.322	0.025	0.019	504	13750	0.010	0.00986
10	312.5	1610	0.430	0.311	0.024	0.014	1000	14030	0.011	0.01246
11	313.6	1627	0.381	0.331	0.028	0.014	1004	13680	0.011	0.01096
12	315.1	1602	0.448	0.329	0.022	0.013	500	13830	0.011	0.00754
13	315.8	1628	0.637	0.373	0.031	0.015	1004	14010	0.011	0.01800
14	313.1	1610	0.372	0.319	0.024	0.014	1000	13780	0.012	0.01168
15	313.0	1619	0.521	0.311	0.039	0.016	494	14280	0.012	0.01413
16	320.9	1638	0.399	0.362	0.032	0.015	1004	14430	0.012	0.00720
17	315.6	1623	0.432	0.338	0.030	0.015	1000	13920	0.012	0.01019
18	314.8	1628	0.346	0.353	0.034	0.016	1494	13270	0.012	0.00678
19	305.8	1632	0.425	0.371	0.038	0.019	1008	13490	0.012	0.00902
20	312.9	1578	0.428	0.296	0.024	0.016	788	13810	0.013	0.00976
21	323.9	1650	0.343	0.344	0.031	0.018	988	13790	0.013	0.00621
22	316.6	1586	0.559	0.334	0.026	0.015	964	13740	0.013	0.01578
23	311.8	1622	0.679	0.340	0.025	0.017	1004	14340	0.013	0.01859
24	315.8	1612	0.484	0.368	0.033	0.016	888	13950	0.013	0.01321
25	314.7	1601	0.470	0.315	0.026	0.015	1008	13760	0.013	0.01099
26	317.7	1644	0.426	0.329	0.037	0.018	1202	13770	0.013	0.00907
27	315.4	1625	0.443	0.332	0.035	0.015	1020	14460	0.014	0.01067
28	314.5	1599	0.507	0.317	0.027	0.021	1036	14010	0.014	0.01654
29	310.0	1622	0.434	0.337	0.031	0.016	990	13450	0.014	0.01522
30	313.1	1618	0.432	0.328	0.029	0.019	998	13970	0.015	0.01548
31	314.3	1578	0.462	0.313	0.018	0.017	494	13790	0.015	0.00897
32	316.5	1618	0.354	0.320	0.019	0.017	1000	13530	0.015	0.01587
33	316.0	1603	0.493	0.322	0.026	0.017	1000	14170	0.015	0.01460
34	314.7	1619	0.379	0.348	0.026	0.021	988	13370	0.015	0.01797

continued..

ble 2.7 (..continued) Test set data and predictions for desulphurization of steel in 300 ton converter

S No -	Metal wt. (ton)	Ini--tial temp. (°C)	Ini--tial wt%C	Ini--tial wt%Mn	Ini--tial wt%P	Ini--tial wt%S	Ore added (Kg.)	Total O_2 blown (m^3)	Actual final wt%S	Predicted final wt%S
35	313.8	1598	0.472	0.340	0.030	0.020	1004	13870	0.015	0.01513
36	312.3	1574	0.532	0.396	0.021	0.018	590	13740	0.016	0.01896
37	315.3	1615	0.434	0.317	0.027	0.019	1010	13870	0.016	0.01436
38	319.9	1609	0.460	0.347	0.031	0.021	494	13810	0.016	0.00914
39	313.4	1608	0.515	0.348	0.029	0.018	1034	14170	0.016	0.01605
40	313.6	1606	0.466	0.336	0.023	0.022	508	13570	0.017	0.01877
41	313.1	1585	0.550	0.304	0.024	0.020	1004	14300	0.017	0.01865
42	315.8	1600	0.383	0.293	0.024	0.019	974	13780	0.017	0.01433
43	316.8	1612	0.288	0.289	0.028	0.022	500	13520	0.017	0.01606
44	315.4	1620	0.358	0.325	0.034	0.019	1028	13620	0.017	0.01456
45	314.7	1626	0.436	0.352	0.034	0.020	1000	13660	0.017	0.01618
46	312.9	1597	0.435	0.317	0.024	0.025	508	14100	0.018	0.00921
47	314.0	1619	0.345	0.323	0.022	0.021	1008	13560	0.018	0.01565
48	314.7	1618	0.389	0.337	0.038	0.021	1014	13680	0.018	0.01574
49	314.0	1595	0.415	0.343	0.022	0.029	500	13510	0.019	0.01814
50	313.9	1619	0.457	0.343	0.027	0.021	834	13850	0.019	0.01820

Data set numbers 7, 13, 16, 18, 21, 23, 26, 31, 38 and 46 were found to be out-laws hence are not lotted in fig.2.11

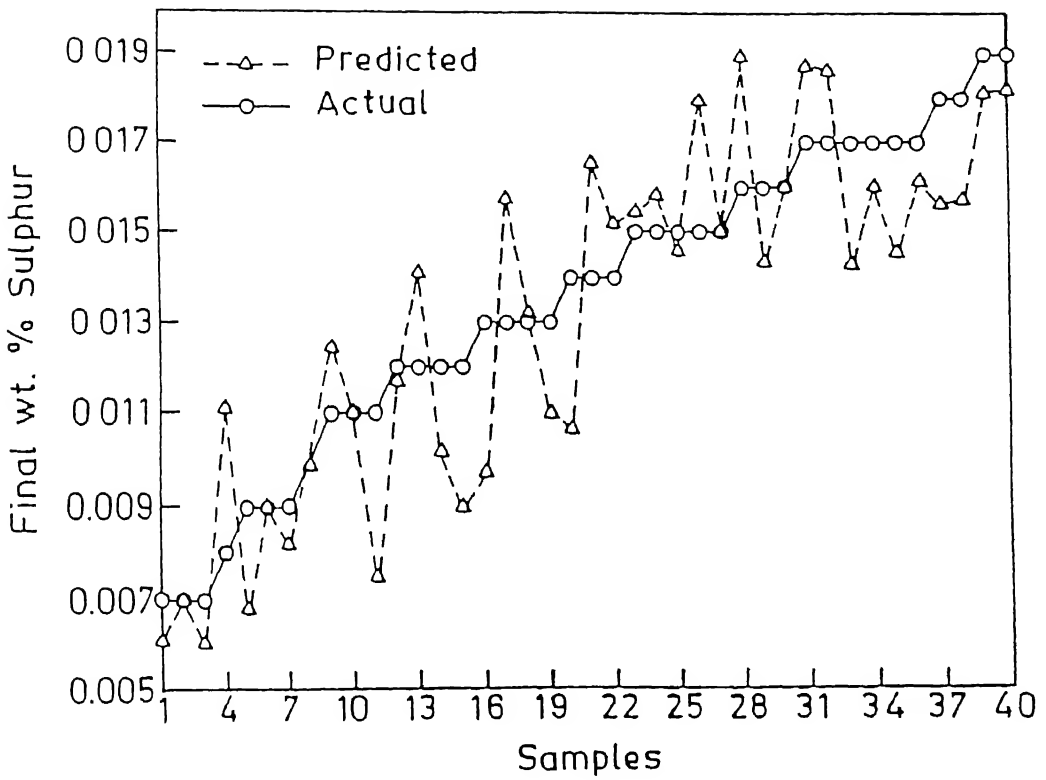


Figure 2.11 Predictions of a trained ANN for desulphurization of steel in combined blown converter.

flurospar added in the bath are not incorporated as input variables. However, both lime and flurospar play a pivotal role in sulphur and phosphorus removal from bath and their incorporation in the input layer is expected to further improve the performance of net, even in the case of outlaws mentioned at the bottom of the Table-2.8. and Table-2.10.

In another study the patterns used for training are taken from the heats without argon bubbling during refining. That means the net is not introduced to the effect of argon bubbling on impurity removal. So when the trained net is tested with the patterns where argon was purged, then also it predicts the final value of phosphorus in the relatively higher range as shown in Fig.(2.13). The result indicates that ANN could not pick up the significance of argon purging, which enhances phosphorus elimination.

2.3 Discussion

The neural network simulation of desulphurization and dephosphorization has shown that generalization performance of ANN is satisfactory, even when trained with very limited number of patterns. The study of effects of various net parameters and net configurations brought out some important points, which are summarized below.

(1) An optimal selection of neurons in hidden layer is an important step. Too many hidden neurons may render the mapping of the patterns difficult (marked by the perturbations in [Fig.2.3]), while too few hidden neuron may act as a hinderance in convergence.

(2) Momentum factor (α) should not be very high. Momentum factor controls the influence of previous iteration on the modification of weights in present iteration. Too much emphasis on the weight changes in previous iteration may slow down the learning process.

Table.2.8 Training patterns for dephosphorization of steel in 300 ton converter

S No -	Metal wt. (ton)	Ini--tial temp. (°C)	Ini--tial wt%C	Ini--tial wt%Mn	Ini--tial wt%P	Ini--tial wt%S	Ore added (Kg.)	Total O_2 blown (m^3)	Final wt%P
1	326.5	1620	0.541	0.354	0.034	0.014	1000	14310	0.015
2	315.0	1612	0.503	0.321	0.032	0.016	988	14390	0.012
3	310.2	1639	0.559	0.368	0.036	0.020	1004	14550	0.014
4	325.5	1608	0.484	0.314	0.024	0.018	1004	14370	0.011
5	315.1	1594	0.448	0.336	0.023	0.018	1008	13520	0.011
6	314.6	1606	0.420	0.338	0.031	0.020	500	13580	0.013
7	315.4	1607	0.525	0.338	0.037	0.028	1024	13870	0.014
8	313.5	1628	0.391	0.343	0.031	0.021	1010	13520	0.013
9	316.7	1629	0.485	0.334	0.037	0.017	1000	13920	0.013
10	314.6	1612	0.537	0.342	0.026	0.017	1500	13630	0.009
11	313.2	1622	0.405	0.358	0.033	0.017	998	14280	0.008
12	315.6	1623	0.463	0.374	0.036	0.019	1000	14190	0.017
13	315.4	1632	0.416	0.344	0.033	0.013	1000	14560	0.013
14	313.7	1657	0.130	0.264	0.015	0.012	598	13690	0.011
15	313.9	1642	0.406	0.372	0.033	0.015	1004	13990	0.012
16	316.1	1652	0.443	0.338	0.031	0.010	998	14020	0.012
17	315.7	1614	0.531	0.347	0.024	0.014	508	13810	0.014
18	317.0	1593	0.701	0.319	0.024	0.017	788	13980	0.014
19	316.5	1591	0.606	0.333	0.032	0.020	500	14220	0.015
20	315.6	1614	0.586	0.341	0.029	0.020	1000	14210	0.013
21	315.8	1616	0.568	0.341	0.029	0.020	1000	14210	0.013
22	315.2	1596	0.654	0.334	0.025	0.019	1008	14090	0.010
23	315.0	1603	0.323	0.252	0.014	0.020	500	13710	0.009
24	314.7	1618	0.515	0.340	0.027	0.015	1002	14100	0.011
25	310.1	1593	0.464	0.300	0.020	0.016	976	13980	0.009
26	313.4	1589	0.506	0.288	0.021	0.016	988	14050	0.008
27	314.0	1592	0.397	0.302	0.021	0.021	1000	14030	0.009
28	313.0	1595	0.291	0.303	0.020	0.018	1000	13820	0.010
29	313.7	1584	0.456	0.297	0.022	0.018	814	14250	0.009
30	314.2	1586	0.433	0.260	0.022	0.018	800	13970	0.010
31	315.6	1623	0.387	0.332	0.028	0.016	1004	14000	0.012
32	315.3	1617	0.357	0.324	0.028	0.015	998	13960	0.013
33	313.9	1632	0.232	0.310	0.026	0.020	498	14000	0.016
34	310.6	1641	0.497	0.348	0.039	0.019	1500	14010	0.015
35	314.7	1615	0.590	0.334	0.034	0.013	1010	14490	0.013

continued..

Table.2.8 (..continued) Training patterns for dephosphorization of steel in 300 ton converter

S No -	Metal wt. (ton)	Ini--tial temp. (°C)	Ini--tial wt%C	Ini--tial wt%Mn	Ini--tial wt%P	Ini--tial wt%S	Ore added (Kg.)	Total O_2 blown (m^3)	Final wt%P
36	315.5	1603	0.463	0.315	0.029	0.016	1000	13610	0.012
37	314.7	1603	0.380	0.260	0.016	0.037	1014	13820	0.007
38	314.6	1611	0.252	0.317	0.019	0.019	1000	13790	0.009
39	313.9	1625	0.299	0.320	0.027	0.017	1014	13680	0.010
40	314.5	1630	0.407	0.322	0.036	0.017	1000	14080	0.013
41	322.5	1633	0.318	0.315	0.039	0.016	1500	13890	0.015
42	316.0	1610	0.308	0.323	0.026	0.009	998	13480	0.012
43	315.2	1610	0.471	0.357	0.034	0.013	1004	13860	0.013
44	315.7	1613	0.313	0.315	0.026	0.019	998	13670	0.012
45	312.9	1629	0.447	0.345	0.032	0.017	1010	14440	0.011
46	314.4	1641	0.442	0.333	0.036	0.025	1002	13780	0.011
47	315.2	1625	0.326	0.329	0.023	0.018	1010	12970	0.016
48	315.0	1614	0.316	0.321	0.023	0.007	508	13880	0.012
49	314.2	1569	0.646	0.336	0.030	0.012	510	14490	0.011
50	305.3	1623	0.446	0.405	0.034	0.017	1000	13230	0.016

Table.2.9 Test set data and predictions for dephosphorization of steel in 300 ton converter

S No -	Metal wt. (ton)	Ini--tial temp. (°C)	Ini--tial wt%C	Ini--tial wt%Mn	Ini--tial wt%P	Ini--tial wt%S	Ore added (Kg.)	Total O_2 blown (m^3)	Actual final wt%P	Predicted final wt%P
1	315.7	1604	0.374	0.329	0.022	0.010	514	13820	0.007	.01050
2	313.6	1606	0.466	0.336	0.023	0.022	508	13570	0.008	.01109
3	312.9	1597	0.435	0.317	0.024	0.025	508	14100	0.008	.01537
4	313.3	1585	0.550	0.304	0.024	0.020	1004	14300	0.008	.00744
5	312.9	1578	0.428	0.296	0.024	0.016	788	13810	0.008	.00836
6	314.4	1578	0.462	0.313	0.018	0.017	494	13790	0.008	.00989
7	313.1	1610	0.372	0.319	0.024	0.014	1000	13780	0.009	.01055
8	314.4	1618	0.425	0.337	0.025	0.008	1000	14110	0.009	.00936
9	314.0	1619	0.345	0.323	0.022	0.021	1008	13560	0.009	.01096
10	314.7	1618	0.389	0.337	0.038	0.021	1014	13680	0.010	.01691
11	316.5	1618	0.354	0.320	0.019	0.017	1000	13530	0.010	.00916
12	314.1	1610	0.485	0.346	0.029	0.009	498	14530	0.010	.01685
13	313.7	1591	0.423	0.317	0.024	0.026	490	14080	0.010	.01478
14	311.8	1622	0.679	0.340	0.025	0.017	1004	14340	0.010	.01561
15	312.5	1610	0.430	0.311	0.024	0.014	1000	14030	0.010	.01340
16	312.3	1574	0.532	0.396	0.021	0.018	590	13740	0.010	.01682
17	315.3	1615	0.434	0.317	0.027	0.019	1010	13870	0.010	.01187
18	315.8	1600	0.383	0.293	0.024	0.019	974	13780	0.011	.01058
19	313.6	1627	0.381	0.331	0.028	0.014	1004	13680	0.011	.01038
20	316.0	1603	0.493	0.322	0.026	0.017	1000	14170	0.011	.00984
21	315.1	1602	0.448	0.329	0.022	0.013	500	13830	0.011	.01132
22	316.6	1586	0.559	0.334	0.026	0.015	964	13740	0.011	.01092
23	316.8	1612	0.288	0.289	0.028	0.022	500	13520	0.011	.00802
24	314.7	1619	0.379	0.348	0.026	0.021	988	13370	0.011	.01483
25	315.8	1612	0.484	0.368	0.033	0.016	888	13950	0.012	.01520
26	314.7	1601	0.470	0.315	0.026	0.015	1008	13760	0.012	.01005
27	313.4	1608	0.515	0.348	0.029	0.018	1034	14170	0.012	.00809
28	313.8	1598	0.472	0.340	0.030	0.020	1004	13870	0.012	.01033
29	314.2	1605	0.519	0.346	0.030	0.023	1004	13930	0.012	.01157
30	314.9	1611	0.423	0.325	0.032	0.012	1014	13690	0.012	.01224
31	315.4	1620	0.358	0.325	0.034	0.019	1028	13620	0.013	.01283
32	313.3	1583	0.466	0.322	0.025	0.019	504	13750	0.013	.01316
33	314.0	1595	0.415	0.343	0.022	0.029	500	13510	0.013	.01175
34	313.9	1619	0.457	0.343	0.027	0.021	834	13850	0.013	.01103
35	314.5	1599	0.507	0.317	0.027	0.021	1036	14010	0.013	.00892

continued..

Table.2.9 (..continued) Test set data and predictions for dephosphorization of steel in 300 ton converter

S No -	Metal wt. (ton)	Ini- -tial temp. (°C)	Ini- -tial wt%C	Ini- -tial wt%Mn	Ini- -tial wt%P	Ini- -tial wt%S	Ore added (Kg.)	Total O_2 blown (m^3)	Actual final wt%P	Predicted final wt%P
36	313.1	1618	0.432	0.328	0.029	0.019	998	13970	0.013	.01198
37	314.9	1611	0.379	0.334	0.028	0.023	500	13420	0.013	.01151
38	320.9	1638	0.399	0.362	0.032	0.015	1004	14430	0.014	.01515
39	314.2	1603	0.512	0.337	0.030	0.008	790	13480	0.014	.01287
40	323.9	1650	0.343	0.344	0.031	0.018	988	13790	0.014	.00915
41	305.8	1632	0.425	0.371	0.038	0.019	1008	13490	0.014	.01700
42	312.9	1624	0.568	0.317	0.027	0.012	1000	13100	0.014	.01540
43	313.0	1619	0.521	0.311	0.039	0.016	494	14280	0.014	.01410
44	313.1	1624	0.633	0.341	0.033	0.012	1004	14260	0.015	.01413
45	315.4	1625	0.443	0.332	0.035	0.015	1020	14460	0.015	.01145
46	317.7	1644	0.426	0.329	0.037	0.018	1202	13770	0.016	.01234
47	315.6	1623	0.432	0.338	0.030	0.015	1000	13920	0.016	.01123
48	314.7	1626	0.436	0.352	0.034	0.020	1000	13660	0.016	.01636
49	319.9	1609	0.460	0.347	0.031	0.021	494	13810	0.017	.01469
50	315.8	1628	0.637	0.373	0.031	0.015	1004	14010	0.017	.01460

Data set numbers 2, 3, 10, 12, 13, 14, 15, 16, 23, 24, 25, 27, 35, 40 and 47 were found to be out-laws hence are not plotted in fig.2.12

BP - Conventional Back-propagation with $\beta = 0.5$.
GSS - β is optimized by Golden-section method.
DSC - β is optimized by DSC method.
LT - With logarithmic tranformation of data.

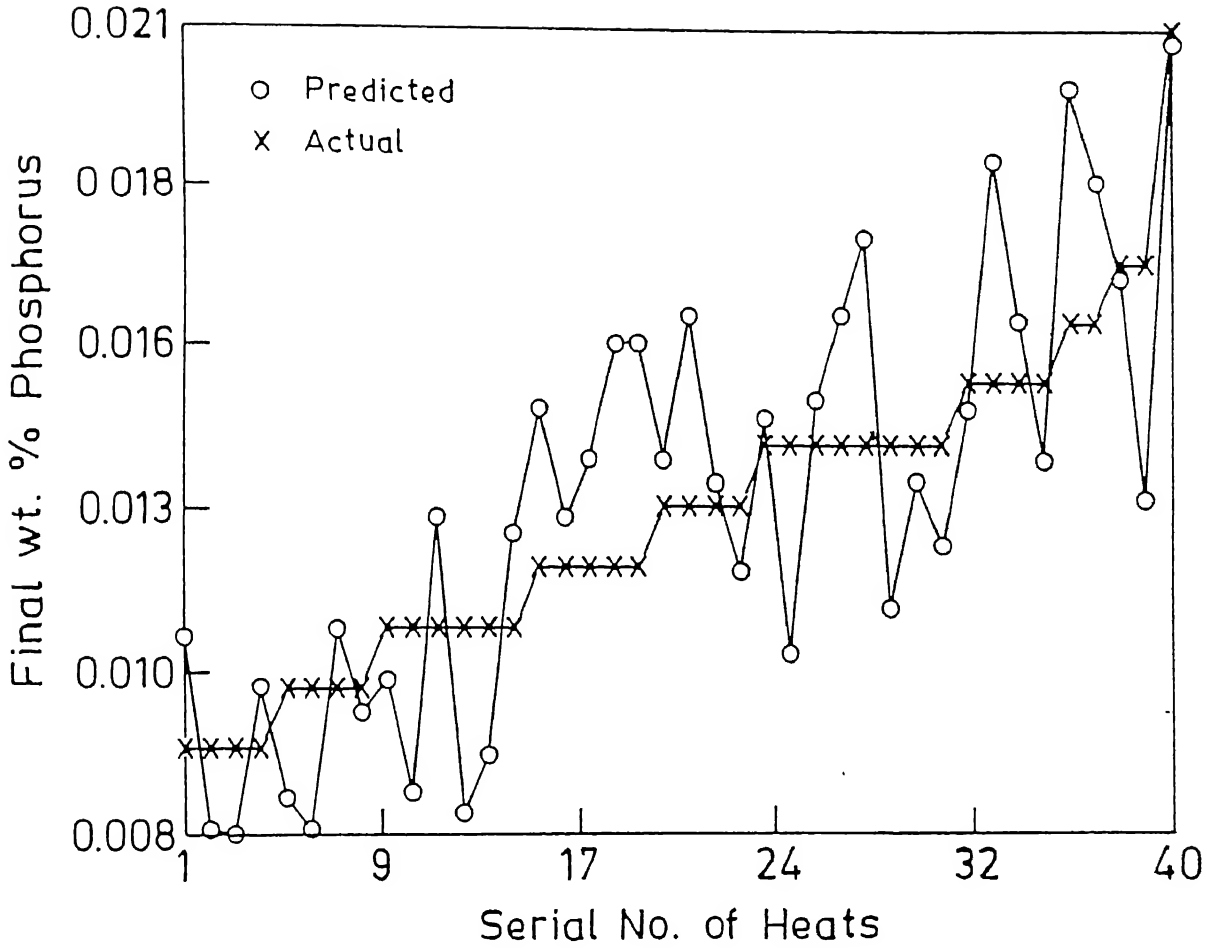


Figure 2.12 Predictions of a trained ANN for dephosphorization of steel in combined blown converter.

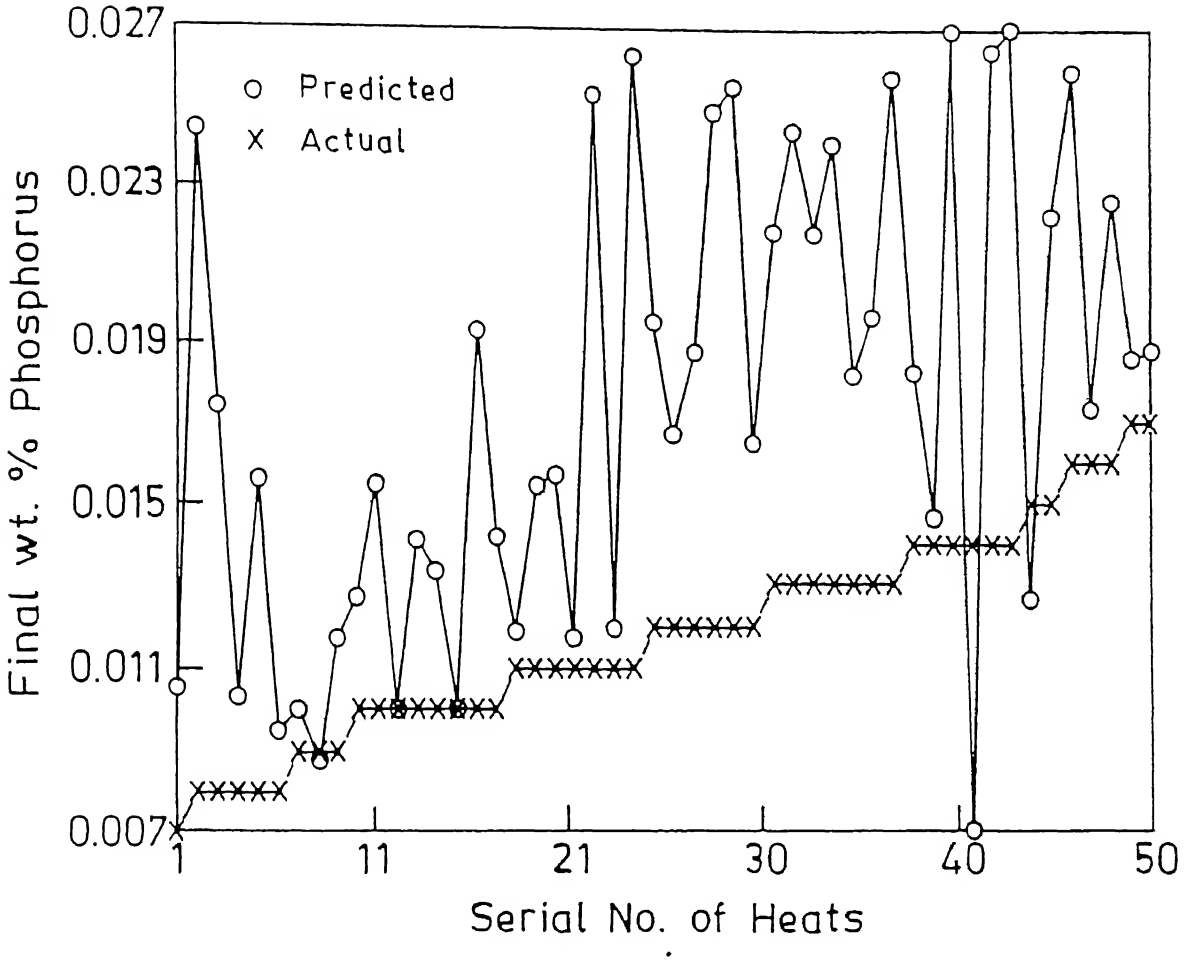


Figure 2.13 Predictions of ANN, trained by data set without argon purging, for de phosphorization of steel in combined blown converter, while test data set are taken from heats with strong argon purging.

(3) Learning rate (β) should also be selected judiciously to obtain low error during training. It is observed that for a complicated net a lower learning rate may give better result.

(4) It is always desirable to feed as many number of patterns as possible. In this way ANN can more easily adapt to the system.

(5) After proper training, the net can be used to find out a particular input, when all other inputs and the target outputs are known and thus the ANN model can function as a versatile means of fault diagnosis and process control.

(6) It is recommended to use a bias node in the net configuration because it takes care of the asymptotic sigmoidal curve, causing better learning of net.

(7) The observation from Fig.(2.13) reveals that ANN is able to adapt to the process well.

(8) In the above mentioned system, different inputs will have different degree of influence on the output. So it is expected that the weights of interconnections between a particular input neuron and all the hidden neurons to be maximum and that of another input to be minimum. But it does not happen. Because of the non-linearity imposed by the sigmoidal function, the influence of that input is not directly discernible.

Chapter 3
Optimization of ANN Model for Hot-metal Desulphurization

The Back-propagation technique [5] using the 'Steepest-descent' method of optimization was used in chapter 2 to develop ANN models for desulphurization and dephosphorization. The independent variable was the weight vector of the interconnections among the neurons and the function to be optimized was the square of the difference of the supervising data and the output of the ANN. In principle, the steepest descent algorithm uses a particular step length of search. In the jargon of ANN this step length is known as learning rate or gain term. In this chapter application of two single dimension techniques, namely Davies-Swann-Campey (DSC) and Golden-section Search (GSS), to optimize this learning rate is discussed. In addition, influence of preprocessing of data by Genetic Adaptive Search (GAS) and logarithmic transformation are discussed so as to find out the best ANN model.

3.1 Optimization of Learning-rate

In the ANN model of desulphurization of iron in 400 ton torpedo vessel, as described in section 2.2 (g), the four inputs used were initial sulphur content of metal (wt%), weight of hot-metal (ton), total weight of powder injected (Kg.) and injection time (min.), respectively and the output was final sulphur content of metal (wt%). A bias node was added to input and hidden layer.

In a single iteration learning rate decides the degree of modification of weights before another pattern is introduced to ANN (Eqn.1.20). Therefore, β decides the value of actual output made by the net for each pattern and has an influence on the root mean squared value of error for each node in the output layer (Eqn.2.2). However, the idea of optimizing learning rate in every iteration is not a feasible one because as β changes the error surface also changes and in the next step of weight adjustment the steepest descent algorithm will find out a new direction. Because of the changing error surface, the algorithm gets confused to find out the right direction of descent. It was decided to first optimize the learning rate and then the weight vector. The root mean squared value of error is an implicit and non-linear function of β. So it is becomes a one dimensional minimization problem. The following two techniques were applied to optimize β. It is assumed that the error (Eqn.2.2) is a unimodal function of β.

3.1.1 Davies-Swann-Campey (DSC) Search Method [14]

DSC algorithm expands the search range exponentially according to

$$\beta_{j+1} = \beta_0 + 2^j.h \qquad 3.1$$

until the relationship of the error $E(\vec{W}, \beta)$ is as follows -

$$E(\vec{W}, \beta_{j+1}) > E(\vec{W}, \beta_j) \qquad 3.2$$

where 'j' is the search times and 'h' is the search step.

After finding out the range where the optimal point lies, it specifies the point by fitting a quadratic curve. The steps of the algorithm are as follows.

Step 1: Initialize weight vector, learning rate and search step as $\vec{W}_0$, β_0 and h. Therefore $E_0 = E(\vec{W}_0, \beta_0)$

Step 2: Calculate $E = E(\vec{W}_0, \beta_1)$, where $\beta_1 = \beta_0 + h..$ If $E_1 > E_0$, then retry with smaller h.

Step 3: Once h is found out, calculate $E_{j+1} = E(\vec{W}_0, \beta_{j+1})$, where $\beta_{j+1} = \beta_0 + 2^j.h.$ It goes on for $j = 1, 2, 3, ...,$ until $E_{j-1} > Ej$.

Step 4: Calculate $E_{(j+1)/2} = E(\vec{W}_0, \beta_{(j+1)/2})$, where $\beta_{(j+1)/2} = (\beta_j + \beta_{j+1})/2$.

Step 5: Define E_1, E_2 and E_3 as the lowest three values among E_{j-1}, E_j, $E_{(j+1)/2}$ and E_{j+1}.

step 6: Draw a quadratic curve through all these points and consider the vertex as the optimum. So the optimal learning rate is calculated by the following equation -

$$\beta_{opt} = \beta_2 - \frac{|\beta_1 - \beta_2|.(E_3 - E_1)}{2(E_3 - 2E_2 + E_1)} \qquad 3.3$$

Once the optimum value of β for that particular initial set of random weight vector is decided the conventional back-propagation is operated to train the ANN. The computer program is given in Appendix-A. This is a powerful algorithm, but with some inherent difficulties. For example, choosing the initial value of learning rate requires some intuition based on the complexity of the problem. The learning rate should lie within the range of 0.0 to 1.0. The algorithm does not have any restriction over this range of β value (0.0 - 1.0). Results of DSC will be discussed later on in the next section.

3.1.2 Golden-section Search (GSS) Method

In the Golden-section Search method, for a given β the r.m.s. value of error (Eqn.2.2) is calculated with respect to several sets of initial random weights for one iteration. Then the average of all these error value is considered as the objective function [f(β)]. In this method the whole search space is mapped in the range of 0.0 to 1.0. Since these are the

lower and the upper bounds of β, in this particular application of Golden-section method mapping is not required . Thereafter a golden number is computed, which comes out to be 0.618. The steps of the algorithm are as follows. [15]

Step 1: Set the lower and the upper bounds of the search space as $x_\beta = 0$ and $y_\beta = 1$, respectively and the range of the search space $L_\beta = y_\beta - x_\beta = 1$. Also set a counter $k = 1$ and decide the required accuracy ϵ.

Step 2: Set $\beta_1 = +(0.618).L_\beta$ and $\beta_2 = y_\beta - (0.618).L_\beta$. Compute $f(\beta_1)$ or $f(\beta_2)$ depending upon whichever is not calculated earlier. Use If $f(\beta_1) < f(\beta_2)$, then the optimum β does not lie from β_2 to y_β. So now the reduced search space is from x_β to β_2. If the reverse occurs then the search space becomes from β_1 to y_β.

Step 3: Set $x_\beta = x_\beta$ and $y_\beta = \beta_2$ or $x_\beta = \beta_1$ and $y_\beta = y_\beta$, depending upon whether $f(\beta_1)$ is less or greater than $f(\beta_2)$. Calculate $L_\beta = y_\beta - x_\beta$. Is $|L_w| < \epsilon$? If no; set $k = k + 1$; go to step 2.

If yes; TERMINATE.

Computer program for GSS method of β optimization is given in Appendix-B.

Again, similar to the DSC method, once the optimum value of β is obtained the conventional back-propagation method is applied, starting from a single set of random weights to train the ANN. In the Golden-section algorithm β is optimized with the help of several sets of random weights and thus an attempt was made to find out a β, suitable for the whole error surface. But similar exercise was not carried out for DSC, because there is no restriction on the upper and lower bounds of β in the DSC algorithm (usually resulting in an erroneous value of β). To obtain a good comparison, the random weight vector from which actual training starts, was kept same for both the cases.

As mentioned earlier the ANN model of desulphurization in 400 ton torpedo vessel

comprised of five nodes (including one bias node) and one output node. The optimum number of hidden neurons is found to be ten (with one bias node). In all, 120 patterns were used to train the ANN (Table 2.5) and later on the trained net was evaluated by 45 sets of data (Table 3.1).

Training curves (error versus number of iterations) are shown in Fig.(3.1); it is clear from that with GSS learning is better than obtained with conventional and DSC method. In the conventional back-propagation method β was kept constant at 0.5. In case of optimization with DSC method the initial value of learning rate was also kept at 0.5 as first guess. The optimum value of learning rate by DSC method is computed to be 0.4997. The errors obtained with back-propagation with fixed learning rate and that with DSC optimized learning rate are nearly same (Fig.3.1). Golden section method yields better results, because it does not require any initial guess value and can be carried out with the help of several sets of random weights. After knowing the optimum value of β by Golden-section (=0.18), DSC method is initialized by $\beta = 0.2$. in this case both the algorithms (DSC and Golden-section search) give almost same results (Fig.3.2). The values of standard deviation (σ) and correlation coefficient (R) for different cases (when a best-fit straight line between the actual and ANN predicted value of final sulphur content (wt%) was drawn through the origin) are given in Table 3.2. It is evident that the ANN trained with Golden-section optimized learning rate possesses the best generalization ability.

3.2 Logarithmic Pre-processing of Data

Thibault *et. al.* [7] suggested that logarithmic transformation of data makes the net

Table 3.1 Test set data and predictions for different training procedure for desulphurization hot-metal in 400 ton torpedo

Serial No. -	Initial wt %S	Metal weight (ton)	CAD added (kg.)	Time (min)	Actual final $wt\%S$	Predicted final $wt\%S$			
						BP	BP+DSC	BP+GSS	BP+GSS+LT
1	0.050	335	1050	25.61	0.009	0.00173	0.0078	0.01021	0.01243
2	0.060	355	1490	38.21	0.013	0.01489	0.0099	0.01012	0.00931
3	0.038	329	740	19.47	0.006	0.00837	0.0143	0.01267	0.01238
4	0.029	356	1020	20.40	0.008	0.01251	0.0020	0.00438	0.00802
5	0.041	355	1672	41.80	0.003	0.00730	0.0031	0.00704	0.00386
6	0.029	340	660	15.71	0.009	0.00208	0.0058	0.00701	0.01326
7	0.036	323	1160	29.74	0.004	0.00355	0.0026	0.01054	0.00843
8	0.035	367	1680	38.18	0.003	0.00310	0.0019	0.00312	0.00343
9	0.030	339	1570	26.61	0.003	0.01176	0.0052	0.00213	0.00171
10	0.028	354	750	18.29	0.012	0.01171	0.0066	0.00537	0.00388
11	0.046	360	1660	36.09	0.007	0.00260	0.0026	0.00262	0.00423
12	0.032	341	730	9.61	0.012	0.00309	0.0063	0.00655	0.01370
13	0.027	341	600	17.65	0.010	0.00691	0.0062	0.00484	0.01164
14	0.031	341	1450	40.28	0.003	0.00101	0.0044	0.00401	0.00027
15	0.022	373	410	8.54	0.011	0.01446	0.0147	0.00961	0.01108
16	0.035	352	2000	47.62	0.003	0.00100	0.0034	0.00228	0.00124
17	0.025	337	530	13.59	0.012	0.00740	0.0057	0.00538	0.01493
18	0.040	343	1540	32.08	0.004	0.01164	0.0035	0.00377	0.00373
19	0.042	343	870	17.06	0.002	0.00165	0.0083	0.00911	0.00741
20	0.049	356	1500	29.41	0.008	0.00285	0.0051	0.00455	0.00537
21	0.065	335	1510	31.46	0.012	0.01414	0.0048	0.01263	0.00740
22	0.062	371	2640	55.00	0.002	0.00294	0.0034	0.00329	0.00525
23	0.050	363	1110	24.13	0.015	0.00153	0.0102	0.00993	0.01178
24	0.030	302	590	13.72	0.009	0.01063	0.0093	0.01349	0.00867
25	0.032	321	1000	22.22	0.006	0.00215	0.0048	0.00743	0.00771
26	0.035	291	660	19.41	0.010	0.01078	0.0056	0.00326	0.00126
27	0.036	303	1380	30.00	0.003	0.00123	0.0032	0.00753	0.00883
28	0.034	318	1330	26.60	0.005	0.00137	0.0023	0.00527	0.00416
29	0.039	320	1595	30.67	0.004	0.00271	0.0032	0.00488	0.00444
30	0.063	299	1500	38.46	0.008	0.01432	0.0000	0.00583	0.00719
31	0.069	322	2167	49.25	0.005	0.01065	0.0084	0.00511	0.00372
32	0.056	313	1030	31.21	0.011	0.01500	0.0024	0.01254	0.00679
33	0.050	340	1080	23.48	0.011	0.00142	0.0078	0.00970	0.01008
34	0.031	325	1330	36.94	0.004	0.00111	0.0027	0.00579	0.00157
35	0.028	360	960	33.10	0.006	0.00232	0.0017	0.00259	0.00250

continued..

Table 3.1 (..continued) Test set data and predictions for different training procedure for desulphurization of hot-metal in 400 ton torpedo

Serial No. -	Initial wt %S	Metal weight (ton)	CAD added (kg.)	Time (min)	Actual final $wt\%S$	Predicted final $wt\%S$			
						BP	BP+DSC	BP+GSS	BP+GSS+LT
36	0.032	340	1830	45.75	0.004	0.00100	0.0032	0.00293	0.00006
37	0.030	339	1130	24.57	0.005	0.00459	0.0031	0.00409	0.00406
38	0.048	330	1258	33.11	0.014	0.00745	0.0096	0.00866	0.01102
39	0.046	319	1390	25.74	0.007	0.01038	0.0097	0.00569	0.00502
40	0.047	337	920	20.44	0.013	0.00151	0.0092	0.01042	0.01005
41	0.059	350	1320	42.58	0.013	0.01500	0.0149	0.01568	0.01261
42	0.029	377	670	19.71	0.011	0.01399	0.0100	0.00942	0.00264
43	0.030	350	710	16.14	0.012	0.01432	0.0077	0.00815	0.00963
44	0.045	331	1135	24.67	0.008	0.00483	0.0109	0.00607	0.00963
45	0.026	362	590	11.57	0.012	0.01485	0.0145	0.00706	0.01489

Legends :

BP - Conventional Back-propagation with $\beta = 0.5$.
GSS - β is optimized by Golden-section method.
DSC - β is optimized by DSC method.
LT - With logarithmic tranformation of data.

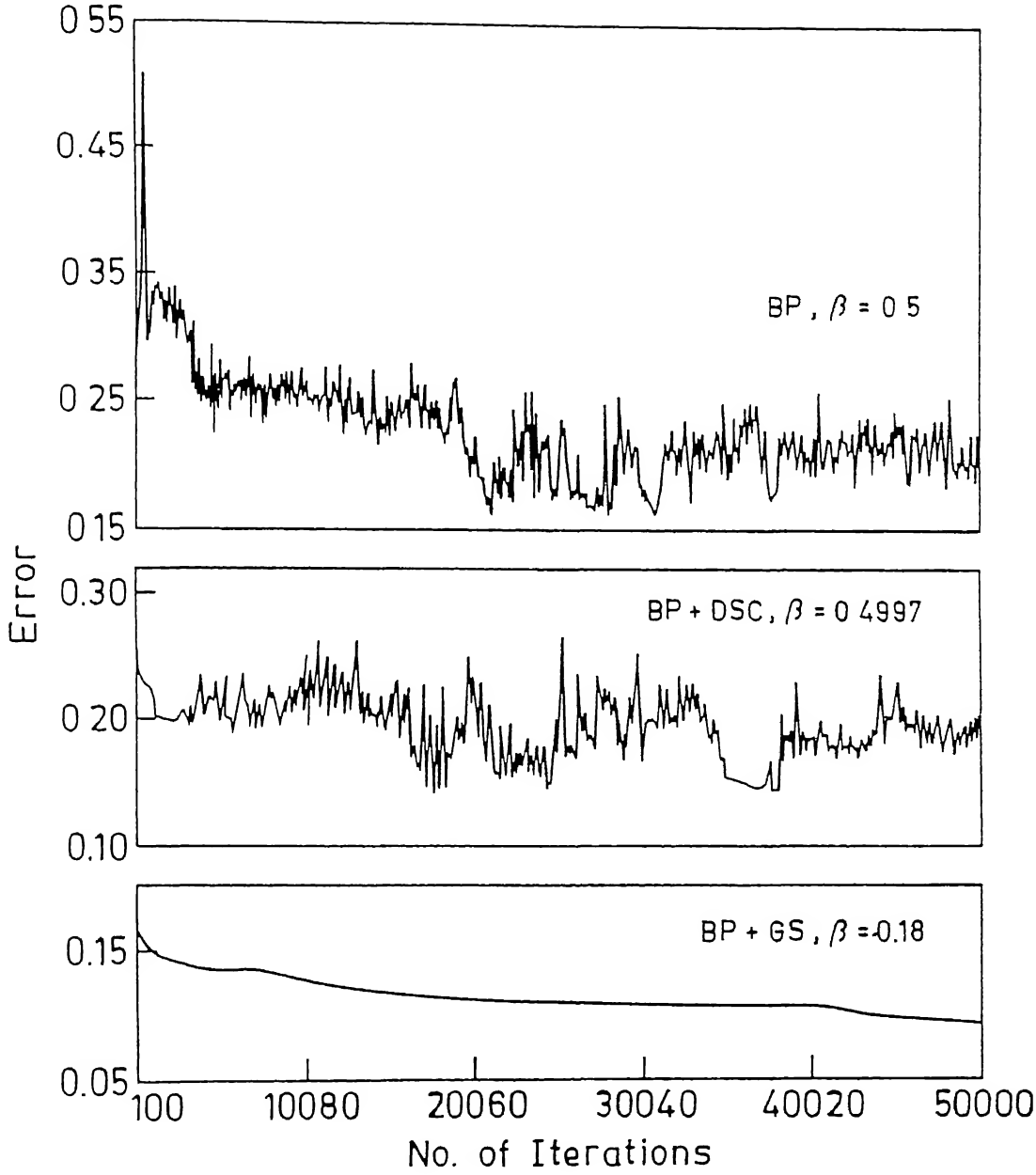


Figure 3.1 Training of [5,10,1] net when learning rate is optimized by different techniques for desulphurization of hot-metal in 400 ton torpedo.

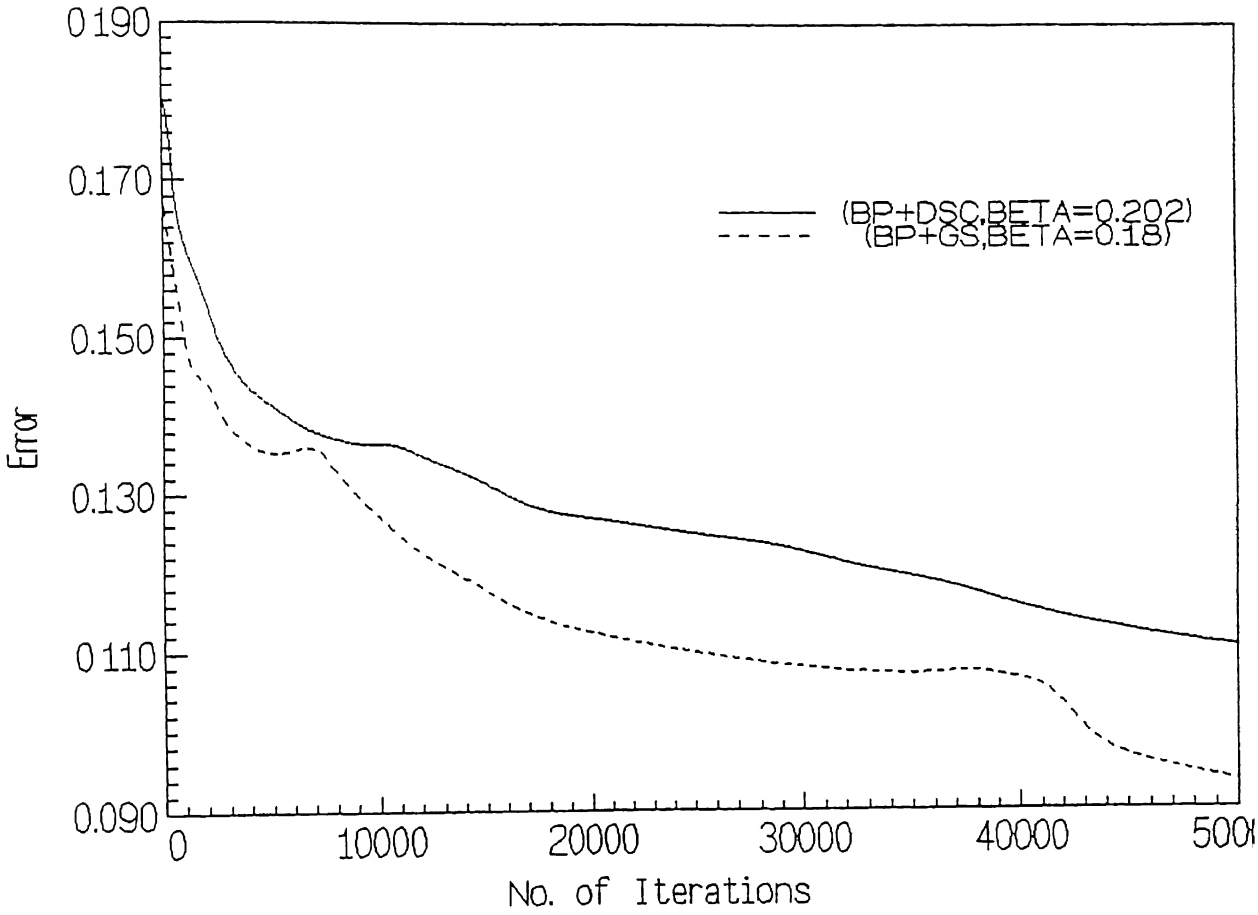


Figure 3.2 Comparison of training of [5,10,1] net when learning rate is optimized by GSS and DSC (with initial $\beta = 0.2$) methods for desulphurization of - hot-metal in 400 ton torpedo.

Table 3.2 Standard deviation (σ) and correlation coefficient (R) on test data when different training procedure are used

Serial No.	cases	σ	R
1	Simple BP	0.005	0.44
2	BP + GSS with no LT	0.003	0.52
3	BP + DSC with no LT (initial $\beta = 0.5$)	0.005	0.28
4	BP + DSC with no LT (initial $\beta = 0.2$)	0.003	0.54
5	BP + GSS with LT	0.003	0.62

Legends :

BP - Conventional Back-propagation with $\beta = 0.5$

GSS - β is optimized by Golden-section method.

DSC - β is optimized by DSC method.

LT - Logarithmic transformation of data.

better conversant with the system. In the desulphurization problem, the numerical values of metal weight (ton), total weight of Calcium Diamide (CAD) used (Kg.) and injection time (min) are of higher magnitude than initial and final sulphur contents of metal (wt%); the initial and final sulphur contents lie between 0.021 to 0.078, whereas hot-metal weight ranges from 280 ton to 408 ton, amount of CAD used ranges from 330 Kg. to 3160 Kg. and the injection time lies between 7.06 to 67.27 min. The later three inputs were transformed logarithmically before scaling so as to compress the range of maximum and minimum value for each variable. It can be seen from Fig.(3.3) and Table 3.2 that when ANN learning rate is optimized by Golden-section method combined with logarithmic preprocessing of data there is not much improvement during the learning phase (Fig.3.3) but the generalization ability of ANN (i.e. predictions during testing) improves (table 3.2).

3.3 Genetic Adaptive Search (GAS) Optimization of ANN

The Eqn.(2.1) can be written as -

$$ln\left(\frac{(\%S)_i}{(\%S)_f}\right) - (C_1.a + C_2.b + C_3.c)\frac{t_s n}{V} = 0 \qquad 3.4$$

where C_1, C_2 and C_3 are constants and a, b and c are defined in section 2.1. Rastogi et. al. [13] applied GAS to optimize C_1, C_2 and C_3 and showed that $C_1 = 1.30$, $C_2 = 0.03$ and $C_3 = 0.0$.

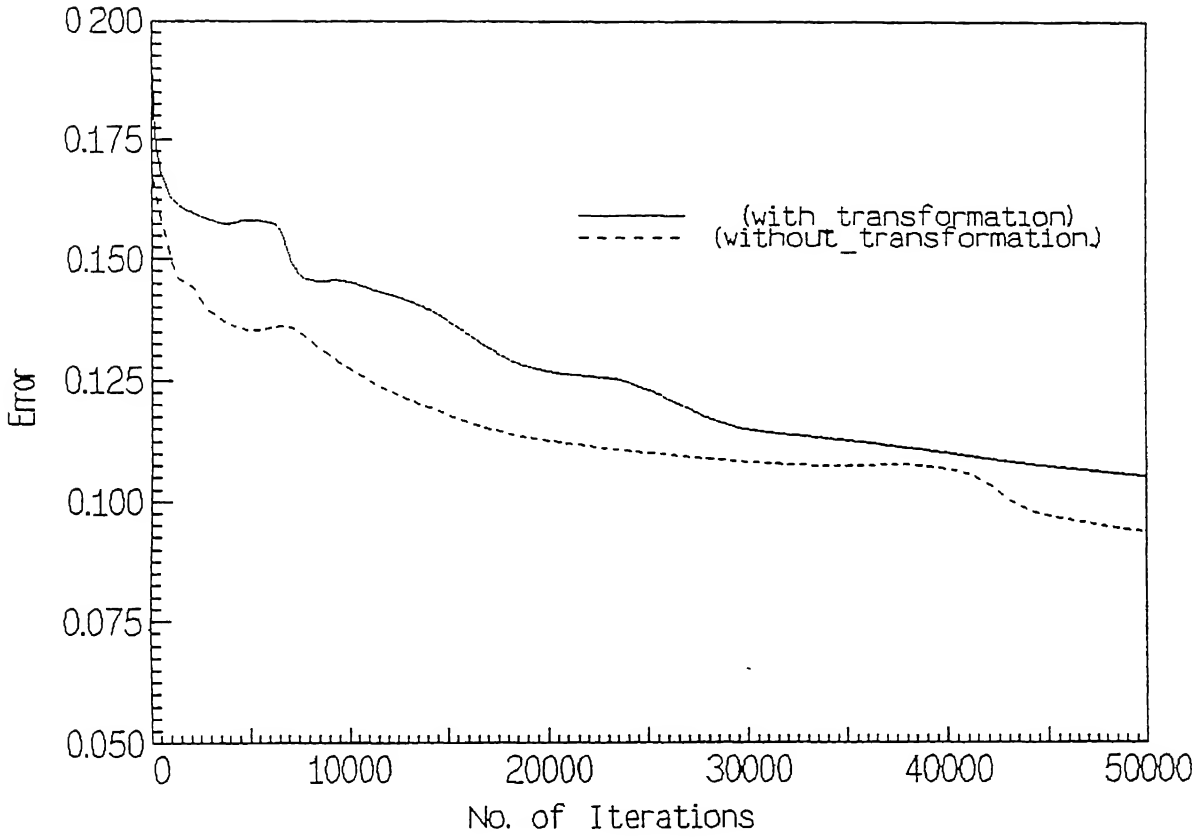


Figure 3.3 Effect of logarithmic transformation of data on training of [5,10,1] net for desulphurization of hot-metal in 400 ton torpedo.

Thus Eqn.(3.4) reduces to-

$$ln\left(\frac{(\%S)_t}{(\%S)_f}\right) = (1.30.a + 0.03.b)\frac{t_in}{V} \qquad 3.5$$

Values of a and b can be calculated from practical data since other parameters given by Eqn.(2.1) are known [13]. Now for each set of data given in Table 2.5 and Table 3.1 all the terms of Eqn.(3.5) are known and summarized in Table 3.3 and Table 3.4, respectively. It can be considered as an ANN structure with two inputs, viz. $1.30.a.(t_in/V)$ and $0.03.b.(t_in/V)$ and one output $ln(S_t/S_f)$. The optimum net architecture was found to be (3,3,1) i.e. it contains 3 nodes each in the input and hidden layers (including one bias neuron) and 1 node in the output layer. In section 3.1 final sulphur contents (wt%) for various test sets were predicted by different types of ANN models (Table 3.1) and from that the actual and predicted values of $ln(S_t/S_f)$ can be calculated. These results are also compared with that of GAS assisted ANN model in Table 3.6. Predictions of various GAS assisted ANN models are shown fig.(3.4).

The performance of GAS assisted ANN models are better than that of ordinary ANN models (Table 3 2 and Table 3.5). However the results obtained by GAS model (by Eqn.3.5) alone are slightly better than that of GAS assisted ANN models. In either cases it can be said that the knowledge of a process model, developed from thermodynamics and kinetics consideration is always an advantage, although with ANN alone one can afford to do without them.

Some important observations of this chapter are summarized below.

1) Training of Adaptive Neural Net improves by addition of bias node in all the layers of neurons except the output layer. Eventually the generalization ability of the net also

Table 3.3 Training patterns for GAS assisted ANN models for desulphurization in 400 ton torpedo

Serial No.	$(1.03.a.\frac{t_{in}}{V})$	$(0.03.b.\frac{t_{in}}{V})$	$ln(S_i/S_f)$
1	.562	.312	1.33500
2	.688	.359	1.18562
3	0.628	0.418	2.07944
4	0.440	0.195	0.767255
5	1.716	0.819	3.06027
6	0.974	0.433	1.76359
7	0.468	0.213	0.767255
8	0.483	0.273	1.01857
9	0.965	0.397	1.73460
10	0.728	0.339	1.52606
11	1.124	0.611	2.19722
12	0.953	0.444	1.86322
13	1.060	0.494	2.03688
14	0.815	0.380	1.17866
15	0.809	0.332	1.04982
16	1.112	0.494	2.14007
17	0.365	0.166	0.780159
18	0.907	0.463	2.07944
19	0.892	0.386	1.27629
20	0.626	0.333	0.9.71861
21	0.847	0.414	2.11021
22	1.329	0.620	1.70475
23	1.502	0.700	2.70805
24	0.883	0.294	0.955511
25	1.164	0.607	1.96166
26	1.744	0.833	2.81341
27	1.120	0.448	2.16905
28	1.150	0.562	2.83321
29	0.753	0.410	2.01490
30	0.592	0.302	0.6.93147
31	0.642	0.314	1.46634
32	0.426	0.199	0.810930
33	2.100	0.909	3.33220
34	1.233	0.698	1.09861
35	0.621	0.262	0.624154
36	0.654	0.370	1.09861
37	0.565	0.314	0.980829
38	1.913	0.935	3.33220
39	0.985	0.481	1.87180
40	0.777	0.414	1.81238

continued..

Table 3.3 (continued..) Training patterns for GAS assisted ANN models for desulphurization in 400 ton torpedo

Serial No.	$(1.03.a.\frac{t_{in}}{V})$	$(0.03.b.\frac{t_{in}}{V})$	$ln(S_i/S_f)$
41	0.727	0.339	1.27297
42	0.870	0.348	2.01490
43	0.548	0.304	1.28093
44	1.371	0.624	3.21888
45	1.095	0.535	2.10006
46	1.199	0.666	2.46385
47	1.465	0.813	1.88031
48	1.343	0.597	1.86322
49	1.335	0.711	2.99573
50	1.302	0.694	3.19867
51	0.281	0.137	0.479573
52	0.790	0.368	1.64223
53	1.280	0.626	2.94444
54	0.837	0.521	2.33537
55	0.747	0.406	1.21640
56	0.968	0.387	1.72277
57	1.195	0.716	1.57554
58	0.694	0.324	0.929536
59	1.062	0.601	2.81341
60	1.192	0.635	1.88031
61	0.823	0.402	1.34707
62	1.353	0.706	1.94591
63	0.813	0.352	1.14862
64	0.313	0.174	0.780159
65	0.591	0.236	1.50408
66	0.864	0.441	2.01490
67	0.700	0.334	1.44692
68	1.392	0.541	1.97408
69	0.840	0.308	1.41707
70	0.661	0.264	1.58045
71	1.048	0.477	1.88273
72	0.542	0.301	0.875469
73	0.738	0.344	1.38629
74	1.127	0.551	2.01490
75	0.763	0.322	1.33500
76	2.139	1.140	2.36085
77	0.305	0.153	0.646627
78	0.893	0.377	1.72277
79	0.403	0.192	1.09861
80	1.053	0.538	2.12823

continued..

Table 3.3 (continued..) Training patterns for GAS assisted ANN models for desulphurization in 400 ton torpedo

Serial No.	$(1.03.a.\frac{t_m}{V})$	$(0.03.b.\frac{t_m}{V})$	$ln(S_i/S_f)$
81	1.428	0.634	1.89085
82	0.982	0.458	1.41099
83	1.072	0.536	2.04769
84	1.355	0.782	2.30258
85	0.548	0.231	1.45529
86	0.225	0.128	1.68640
87	0.515	0.252	1.48160
88	1.267	0.732	2.86220
89	1.689	0.825	2.26176
90	0.577	0.288	1.12847
91	1.265	0.477	2.05412
92	0.278	0.120	1.68176
93	0.964	0.589	2.04307
94	1.908	0.975	2.44235
95	0.307	0.143	1.46634
96	0.420	0.196	0.733969
97	0.791	0.378	1.85630
98	1.648	0.988	2.26868
99	0.349	0.198	1.09861
100	1.405	0.624	2.11021
101	0.895	0.606	1.97408
102	1.538	0.649	1.79176
103	1.282	0.626	2.61496
104	1.463	0.731	2.94444
105	1.768	0.785	3.02852
106	1.391	0.788	2.87168
107	1.120	0.485	1.48160
108	0.446	0.252	0.969401
109	1.029	0.514	2.15948
110	0.651	0.369	1.25276
111	0.952	0.360	1.14210
112	1.497	0.665	1.20397
113	0.741	0.354	1.83828
114	1.344	0.492	2.60269
115	0.354	0.173	0.733969
116	0.587	0.274	1.52606
117	0.792	0.369	1.15745
118	0.614	0.368	1.68640
119	0.925	0.421	2.07944
120	0.957	0.287	1.85630

Table 3.4 Patterns for testing the GA assisted ANN model and test results for desulphuriza in 400 ton torpedo.

Sr. No.	$(1.03.a.\frac{l_{in}}{V})$	$(0.03.b.\frac{l_{in}}{V})$	Actual $ln(S_i/S_f)$	Predicted $ln(S_i/S_f)$			
-				GAS+ANN ($\beta = 0.5$)	GAS+ANN (DSC)	GAS+ANN (GSS)	GAS(only (Eqn.3.5
1	0.835	0.374	1.71480	1.92941	1.9148	2.03746	1.30411
2	1.175	0.501	1.52940	2.39689	2.3556	2.27919	1.79303
3	0.646	0.268	1.84583	1.58813	1.6220	1.68108	0.974374
4	0.626	0.342	1.28785	1.70635	1.7197	1.85553	1.08022
5	1.286	0.562	2.61496	2.52497	2.4911	2.32895	1.98519
6	0.505	0.232	1.17007	1.45023	1.5036	1.52396	0.797641
7	1.005	0.429	2.19722	2.16828	2.1320	2.18179	1.53421
8	1.136	0.546	2.45674	2.43773	2.3960	2.27992	1.83692
9	0.857	0.553	2.30259	2.28694	2.2383	2.22904	1.62038
10	0.564	0.253	0.84730	1.51332	1.5576	1.59864	0.881509
11	1.095	0.550	1.88273	2.42193	2.3787	2.27109	1.80981
12	0.308	0.256	0.98083	1.38416	1.4442	1.48278	0.676804
13	0.565	0.210	0.99325	1.45188	1.5063	1.50882	0.810691
14	1.290	0.508	2.33537	2.46515	2.4282	2.32414	1.89736
15	0.250	0.131	0.69315	1.24623	1.3259	1.32126	0.422564
16	1.477	0.678	2.45674	2.67199	2.6595	2.41695	2.33469
17	0.440	0.188	0.73397	1.36479	1.4304	1.42311	0.671856
18	1.021	0.536	2.30258	2.36089	2.3154	2.25126	1.72601
19	0.543	0.303	3.04452	1.58003	1.6125	1.70760	0.947434
20	0.902	0.503	1.81238	2.23068	2.1863	2.21079	1.57386
21	1.026	0.538	1.68948	2.36680	2.3214	2.25297	1.73281
22	1.619	0.849	3.43399	2.75413	2.7613	2.48516	2.73557
23	0.726	0.365	1.20397	1.82616	1.8232	1.96470	1.20017
24	0.496	0.233	1.20397	1.44687	1.5006	1.52175	0.793141
25	0.756	0.372	1.67398	1.86348	1.8558	1.99474	1.23609
26	0.728	0.271	1.25276	1.64845	1.6745	1.73852	1.04498
27	1.081	0.544	2.48491	2.40624	2.3624	2.26643	1.78757
28	0.913	0.499	1.91692	2.23113	2.1871	2.21069	1.57683
29	1.047	0.595	2.27727	2.45644	2.4130	2.27112	1.84509
30	1.405	0.599	2.06369	2.60128	2.5771	2.38341	2.14314

continued..

Table 3.4 (..contunued) Patterns for testing the GA assisted ANN model and test results for phurization in 400 ton torpedo.

Sr. No.	$(1.03.a.\frac{t_{in}}{V})$	$(0.03.b.\frac{t_{in}}{V})$	Actual $ln(S_i/S_f)$	Predicted $ln(S_i/S_f)$			
-				GAS+ANN ($\beta = 0.5$)	GAS+ANN (DSC)	GAS+ANN (GSS)	GAS(on (Eqn.3.
31	1.670	0.803	2.62467	2.74782	2.7536	2.53785	2.7005
32	1.089	0.393	1.62746	2.16170	2.1288	2.17876	1.5422
33	0.754	0.379	1.51413	1.87604	1.8664	2.00766	1.2467
34	1.241	0.488	2.04769	2.41439	2.3751	2.29912	1.8260
35	1.004	0.318	1.54045	1.94844	1.9370	2.02190	1.3489
36	1.469	0.642	2.07944	2.64923	2.6327	2.41653	2.2686
37	0.791	0.398	1.79176	1.94358	1.9255	2.05941	1.3082
38	1.096	0.455	1.23214	2.27760	2.2369	2.23263	1.6514
39	0.881	0.520	1.88273	2.24633	2.2004	2.21662	1.5853
40	0.662	0.326	1.28520	1.70246	1.7175	1.83720	1.0832
41	1.328	0.450	1.51259	2.40839	2.3714	2.32932	1.8331
42	0.571	0.212	0.96940	1.45781	1.5114	1.51542	0.81882
43	0.504	0.242	0.91629	1.46347	1.5147	1.54380	0.81402
44	0.814	0.409	1.72722	1.98393	1.9612	2.08675	1.3458
45	0.349	0.195	0.77319	1.33471	1.4033	1.40372	0.60879

Legends :

GAS - Genetic Adaptive Search.

ANN - Adaptive Neural Net.

DSC - β is optimized by DSC method.

GSS - β is optimized by GSS methd.

Table 3.5 Standard deviation (σ) and correlation coefficient (R) on test data to predict $ln(S_i/S_f)$, with different training procedure.

Serial No.	Type of training	σ	R
1	ANN with GSS	0.55	0.57
2	ANN with DSC	0.96	0.63
3	ANN with $\beta = 0.5$	0.90	0.48
4	GA+ANN, with GSS	0.51	0.75
5	GA+ANN, with DSC	0.48	0.77
6	GA+ANN, with $\beta = 0.5$	0.48	0.76
7	GA only (eqn.3.5)	0.50	0.78

Legends :

ANN - Adaptive Neural Net.

GSS - β is optimized by Golden-section method.

DSC - β is optimized by DSC method.

GA - Genetic Algorithm.

improves.

2) By dint of Golden-section search technique, relatively accurate determination of learning rate is possible. Thus the erroneous and time consuming steps of trial and error method can be omitted.

3) The assumption that r.m.s. value is a unimodal function of β, appears to be correct, because during Golden-section optimization of β, it was observed that objective function had almost equal and high values at both the ends of the search space. Later on this value decreased in the interior zone.

4) Logarithmic transformation contracts the maximum and minimum range of the training data of high magnitude. Thereafter when they are introduced to ANN, it can make correlation among them with relative ease. Consequently performance of ANN improves in testing phase.

5) If mathematical model of a particular process exists then Genetic Algorithm (GA) is a better option to optimize the model and use for simulation. In these cases GA assisted ANN model does not give better simulation results.

6) ANN model, when supported by mathematical model(s), optimized by Genetic Algorithm, can make correlations between inputs and outputs very easily. That is why GAS assisted ANN model gives better results than when predictions are made with only ANN.

7) In the study mentioned in section 3.4 the ANN has virtually 2 inputs and one output. In this relatively simple structure, there is not much difference in the performance of ANN using conventional BP (Back-propagation) and BP coupled with learning rate optimization techniques. But in a complicated architecture Back-propagation ANN may give considerably better results when β is optimized by Golden-section (Table 3.2).

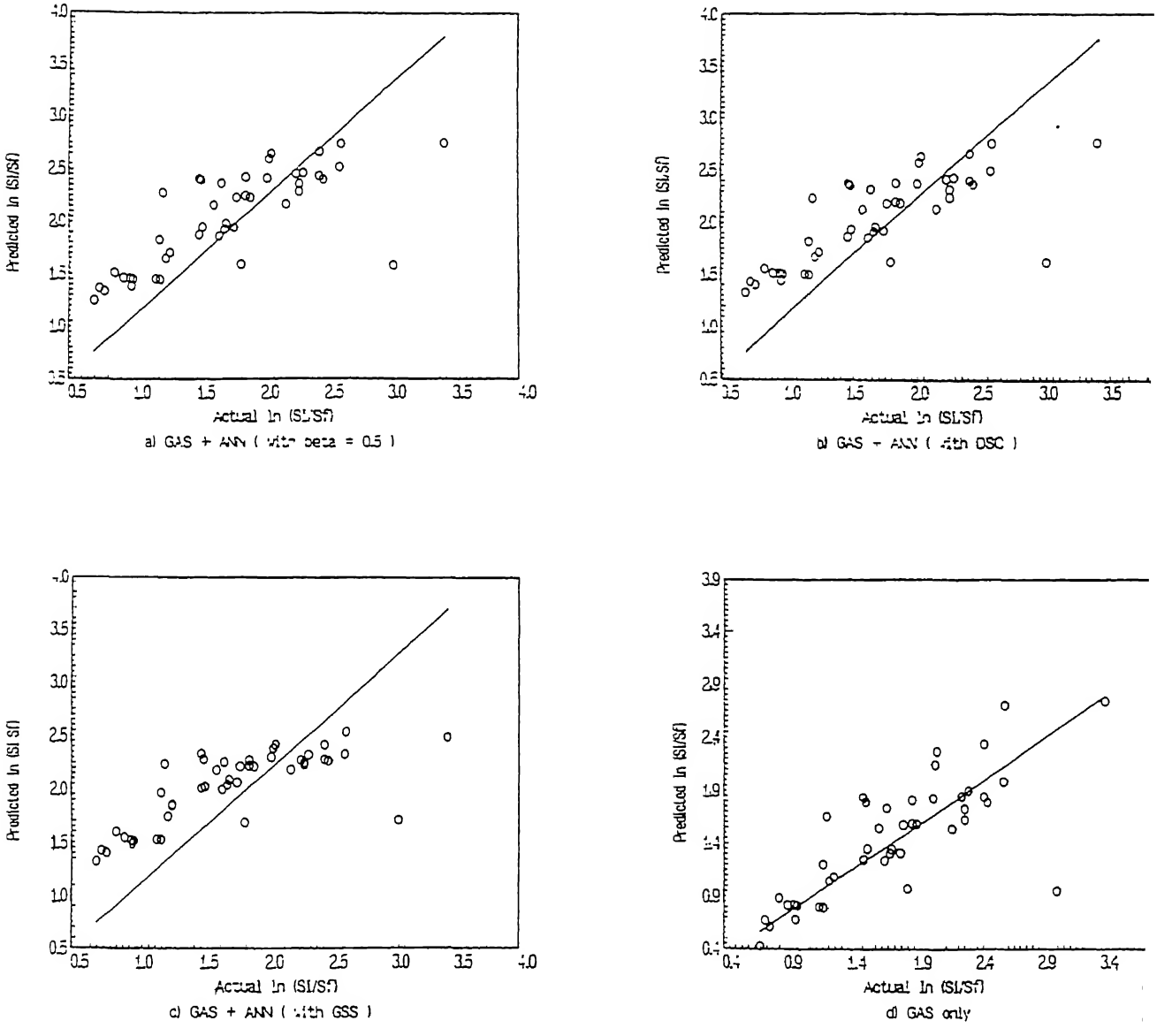


Figure 3.4 Predictions of GAS assisted ANN models for desulphurization in 400 ton torpedo.

Chapter 4
Applicability and Reliability of ANN

Close control of the amounts of sulphur and phosphorus present in hot-metal and steel is a very difficult task for steel-makers. In our study it was observed that ANN can predict the final sulphur and phosphorus content of liquid metal as a function of input variables within the tolerable limits in most of the cases.

Training of ANN is the most crucial and time consuming part of the ANN simulation. The nature and the number of training data should suffice so as to represent the whole process. The number of training data used in our study was not sufficient in some of the cases, even though results obtained are quite satisfactory. Moreover, ANN has a tendency to forget the seldom seen data. So the supervising data should lie as uniformly as possible within the maximum and the minimum range of data. A lot of research is needed to study the statistical nature of the training data.

ANN requires more time for training as the number of training data is increased. One should be a bit cautious regarding the over-learning of net. If the net is trained for more than the optimum number of iterations then it may tend to pick up the very specific nature of supervising data and eventually loose the generalization ability.

In industrial applications the supervising data are continuously updated. Thus, the learning becomes dynamic and helps ANN to get tuned with the recent trends of the process. One such technique has been discussed in section (1.2.1).

The training of ANN can be treated as an optimization problem of very special kind. If the training is not initialized with the right set of initial random weights, then it is possible that the training may stop at a local optima in the error surface. Momentum factor (α) is

introduced to tackle this problem. In the present work Golden-section search technique for single dimensional function optimization has been utilized to find out an optimum value of learning rate (β) with respect to several sets of random weights. Thus an attempt was made to find out a learning rate with some global perspective.

From this study it is clear that the reliability of ANN is not beyond question. A lot of experiments need to be carried out to find out a suitable configuration of ANN so as to simulate processes in iron and steel making. In spite of this, the advantage with ANN is that they can handle very obscure situations, which are otherwise difficult (or impossible) to model theoretically.

References

1) Melsa, P.J.W. : "Neural Network: A Conceptual Overview", A Report of Tellabs Research Center, TRC-89-08, August 1989, Mishawaka, Indiana 46545.

2) Kalra, P.K.; Srivastava, S.C. : Short term course on Neural Network (conducted at I.I.T. Kanpur)

3) Wasserman, P.D. : "Neural Computing: Theory and Practice"ANZA Research Inc. Van Nostrand Reinhold, New York.

4) Lipppman, R.P. : IEEE ASSP Magazine, April 1987 pp 5 - 22.

5) Rumelhart, D.E.; McClelland, J.E. : "Parallel Distributed Processing"UHI, MIT Press, Cambridge,1986, Vol.-1.

6) Anderson, K.; Cook, G.E.; Karsai, G.; Ramaswami, K. : IEEE Trans. on Industry Application Vol. 26, No. 5, 1990, pp 824 - 830.

7) Thibault, J.; Bernard, P.A.G. : Int. Jl. of Heat-Mass Transfer, Vol 34, No. 8 (1991), pp 2063 - 2070.

8) Reuter, M.A.; Van Der Walt, T.J.; Van Denter, J.S.J. : Met. Trans. B, Vol 23B, 1992, pp 643 - 650.

9) Altman, R.; Kellog, H.H. : Trans. Min. Metall. 1972 No. 9, pp C163 - C175.

10) Rankin, W.J.; Met. Trans. B, Vol 17B, 1986, pp 61 - 68.

11) Battle, T.P.; Hager J.P. : Met. Trans. B, Vol 24B, 1990, pp 501 - 510

12) Staib, W.E.; Bliss, N.G.; Staib, R.B. : Iron and Steel Engineer, June 1992, pp 29 - 32.

13) Deo, B.; Boom, R.; : "Fundamentals of Steelmaking Metallurgy", Prentice Hall Int. London, U.K. pp 254 - 261.

14) Takagi, H.; Sakaue, S.; : Systems and Computers in Japan, Vol. 23, No. 1 (1992), pp 101 - 111.

15) Rao, S.S. : Optimization theory and application, Wiley Eastern Ltd., pp 231 - 233.

16) Rastogi, R.; Deo, B.; Deb, K.; Boom, R. : (to be published in Steel Research).